Fuzzy variants of prototype based clustering and classification algorithms

Geweniger, Tina

Fuzzy Variants of Prototype Based Clustering and Classification Algorithms

Tina Geweniger

RIJKSUNIVERSITEIT GRONINGEN

Fuzzy Variants of Prototype Based Clustering and Classification Algorithms

Proefschrift

ter verkrijging van het doctoraat in de
Wiskunde en Natuurwetenschappen
aan de Rijksuniversiteit Groningen
op gezag van de
Rector Magnificus, dr. E. Sterken,
in het openbaar te verdedigen op
maandag 22 oktober 2012
om 14:30 uur

door

Tina Geweniger

geboren op 13 maart 1976
te Altenburg, Duitsland

Promotores:	Prof. dr. M. Biehl
	Prof. dr. T. Villmann
Beoordelingscommissie:	Prof. dr. T. Martinetz
	Prof. dr. M. Opper
	Prof. dr. F. Rossi

ISBN: 978-90-367-5749-2

Acknowledgments

Now, that all the work is done, there is a number of friends and collegues I wish to thank for their support, friendship, understanding, patience and their mere being there whenever needed. Thanks to (in reverse alphabetical order):

— Dietlind Zühlke —
diss-sis: we got through it together

— Prof. Thomas "Villy" Villmann —
thesis advisor: taught me to write scientifically and is a source of new ideas

— Frank-Michael Schleif —
last minute proofreader: useful hints for final version

— MESO Company —
employer: time off for writing my thesis

— Wouter Lueks —
translator: without him there would not be a Dutch summary

— Marika Kästner —
little diss-sis: proofreading, useful hints, and discussions

— Klaus Geweniger —
husband: he believed in me

— Prof. Michael Biehl —
"external" thesis advisor: fighting the dutch bureaucracy and excellent cooking

— all members of the Mittweida–Bielefeld–Groningen research group —
many ideas, discussions, and lots of fun ☺

Contents

Abbreviations and Symbols **ix**

1 Introduction **1**
1.1 Scope . . . 2
1.2 Outline . . . 4

2 General notes about clustering and classification **5**
2.1 Vector Quantization and Clustering . . . 5
2.1.1 Vector Quantization . . . 6
2.1.2 c-Means . . . 8
2.1.3 Self-Organizing Maps . . . 11
2.1.4 Neural Gas . . . 13
2.1.5 Fuzzy c-Means . . . 15
2.1.6 Median c-Means . . . 16
2.1.7 Affinity Propagation . . . 17
2.2 Prototype Based Classification . . . 19
2.2.1 Learning Vector Quantization – LVQ1 . . . 21
2.2.2 Learning Vector Quantization – LVQ2.1 . . . 23
2.2.3 Generalized LVQ . . . 23
2.2.4 Robust Soft LVQ . . . 26
2.2.5 Soft Nearest Prototype Classification . . . 27
2.2.6 Fuzzy Soft Nearest Prototype Classification . . . 29
2.3 Distance measures . . . 30
2.3.1 Euclidean metric and variants . . . 31
2.3.2 Divergences . . . 32
2.3.3 Kernel distances . . . 35

2.3.4 Normalized Information Distance . . . 36
2.4 Validation measures for fuzzy clustering and classification . . . 36
2.4.1 Measures based on separation and compactness . . . 37
2.4.2 Rand index and related measures . . . 38
2.4.3 Cohen' Kappa and Fleiss' Kappa . . . 40

3 Fuzzy clustering for non-standard metrics 45
3.1 Relevance clustering with Fuzzy c-Means . . . 48
3.1.1 Incorporating a relevance parameter into FCM . . . 49
3.1.2 Divergences as dissimilarity measures . . . 50
3.1.3 Proof of concept . . . 54
3.1.4 Real world example – remote sensing data . . . 56
3.1.5 Conclusion . . . 62
3.2 Fuzzy median clustering . . . 62
3.2.1 Median Fuzzy c-Means . . . 63
3.2.2 Fuzzy interpretation of Affinity Propagation . . . 64
3.2.3 Artificial example – overlapping Gaussian distributions . . . 66
3.2.4 Real world example – psychological therapy transcripts . . . 69
3.2.5 Conclusion . . . 71

4 Fuzzy classification 73
4.1 Supervised learning . . . 76
4.1.1 Classifying fuzzy labeled data with FSNPC . . . 76
4.1.2 Fuzzy Robust Soft LVQ . . . 78
4.1.3 Artificial example – overlapping Gaussian distributions . . . 82
4.1.4 Real world example – barlay grain tissue sections . . . 85
4.1.5 Conclusion . . . 88
4.2 Semi–supervised learning . . . 89
4.2.1 Fuzzy Labeled NG . . . 90
4.2.2 Fuzzy Labeled SOM . . . 98
4.A Derivation of the F-FSNPC window rule . . . 102
4.B Derivation of the general FRSLVQ update rule . . . 104

5 Summary 109

Publications 115

Bibliography 119

Nederlandse samenvatting 131

Abbreviations and Symbols

$\boldsymbol{v} \in \mathbb{R}^d$	d-dimensional input vector (resp. data sample)
$\boldsymbol{w} \in \mathbb{R}^d$	d-dimensional prototype (resp. codebook vector, reference vector, exemplar)
$d(\boldsymbol{v}, \boldsymbol{w})$	general distance measure or dissimilarity
$E_{xyz}(...)$	cost function for algorithm XYZ
N_C	Number of classes or clusters
N_P	Number of prototypes (resp. codebook vectors, reference vectors, exemplars)
N_V	Number of input vectors (resp. data samples)
$P(V)$	data distribution in the input space
$u_j(\boldsymbol{v}_i)$	Membership assignment of $\boldsymbol{v}_i$ to prototype $\boldsymbol{w}_j$
V	Set of input vectors with $V = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{N_V}\} \subset \mathbb{R}^d$
W	Set of output vectors with $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{N_P}\} \subset \mathbb{R}^d$

ANN	Artificial Neural Network
CM	c-Means
FAP	Fuzzy Affinity Propagation
FCM	Fuzzy c-Means
FLNG	Fuzzy Labeled Neural Gas
FLSOM	Fuzzy Labeled Self-Organizing Map
FRSLVQ	Fuzzy Robust Soft Learning Vector Quantization
FSNPC	Fuzzy Soft Nearest Prototype Classifier
FSOM	Fuzzy Self-Organizing Map
GLVQ	Generalized Learning Vector Quantization
LVQ	Learning Vector Quantization
M-CM	Median c-Means

M-FCM	Median Fuzzy c-Means
NG	Neural Gas
NID	Normalized Information Distance
NPC	Nearest Prototype Classifier
R-FCM	Relevance Fuzzy c-Means
RSLVQ	Robust Soft Learning Vector Quantization
SNPC	Soft Nearest Prototype Classifier
SOM	Self-Organizing Map
SVM	Support Vector Machine
VQ	Vector Quantization

Chapter 1

Introduction

Prototype based clustering and classification is one specific topic in the field of machine learning and artificial neural networks (ANN). Machine learning in general is a branch of artificial intelligence and covers the whole field of algorithms which acquire knowledge based on experience. Thereby, it can be distinguished between symbolic learning and learning from examples. Symbolic learning is based on explicit rules in combination with data samples and is also known as inductive logic programming. Algorithms modulating neural networks rely on data samples and a special type-dependend structure, and store the obtained knowledge implicitly within their specific structure.

To the family of ANNs belong graph based methods like Bayesian networks and Markov Random Fields, Spectral Clustering, and prototype based algorithms. In the following chapters the focus is laid on prototype based methods, especially those, where the prototypes are located within the class centers. Non class typical classifiers like Support Vector Machines (SVM) are not within the scope of this thesis.

There are three different learning paradigms to distinguish: unsupervised learning, supervised learning and reinforcement learning. *Unsupervised learning*, in particular clustering, groups data into sets of similar objects and is applicable for explorative data mining, statistical data analysis, pattern recognition, and information retrieval to name just a few. There exists a variety of prototype based algorithms like c-Means (CM) (Ball and Hall 1967, MacQueen 1967), Fuzzy c-Means (FCM) (Dunn 1973, Bezdek 1980, Hathaway and Bezdek 1986), Self-Organizing Maps (SOM) (Kohonen1990), Neural Gas (NG) (Martinetz et al. 1993), and others. The difference to supervised learning or classification is, that the data samples used to train the artificial neural network are not labeled. Therefore, these algorithms are suitable dealing with data, where no class information is given. The algorithms allow a first inspection of the data and help to discover hidden structures. In the machine learning context these groups of similar objects are called clusters.

Supervised learning refers to algorithms, where during the training process available class information of the data samples is taken into account. This class information is usually obtained with the help of domain experts. Based on the learned structure, it is possible to classify unknown data samples. For prototype based clas-

sifiers new data samples are assigned to the class of the closest prototype respectively class center or class typical representant. Well known examples are Learning Vector Quantization (LVQ) (Kohonen 1986) and variants thereof (Sato and Yamada 1996, Seo and Obermayer 2003), and other Nearest Prototype Classifiers (NPC).

The third learning paradigm *reinforcement learning* is not covered within this thesis. Rather than learning immediately through examples, it requires an external feedback or signal which measures the current state of the system. However, the reward is not given instantanuosly, but with a time delay with respect to a certain action, such that a given signal responses to an action many timesteps before. Based on this signal, the actions are adapted to gain maximum long term reward. (Sutton 1984)

In practical applications, data, which in fact belongs to different groups, i. e. clusters or classes, might be overlapping and therefore cannot be separated clearly. For this kind of data particular supervised and unsupervised methods have been developed, e. g. Fuzzy c-Means (FCM) (Dunn 1973, Bezdek 1980, Hathaway and Bezdek 1986), Fuzzy SOM (FSOM) (Bezdek et al. 1992), and Fuzzy Soft Nearest Prototype Classification (FSNPC) (Villmann, Schleif and Hammer 2006). There, the data points, or for FSNPC the prototypes, are partially assigned to the clusters respectively classes reflecting the uncertainty in the data and allowing insecure decisions.

These overlapping data sets are referred to as *fuzzy data* and the respective learning schemes are *fuzzy methods*. Here, the term *fuzzy* refers to probabilistic or possibilistic assignments[1] of data points to clusters or classes and has to be distinguished from fuzzy sets or fuzzy logic. In the context of machine learning it is *learning with uncertainties* and, accordingly, fuzzy clustering respectively fuzzy classification can be understood as probabilistic or possibilistic clustering or classification.

1.1 Scope

The main theme of this thesis is to extend known prototype based methods for clustering and classification to handle fuzzy data. Since clustering and classification are two different learning paradigms with a slightly different meaning concerning fuzziness, they are treated separately. While a cluster solution is called fuzzy if the data belonging to different clusters are overlapping, a fuzzy classification is either based on overlapping classes or on fuzzy labeled training data or both, where the fuzzy class assignments can also be interpreted as assignment probabilities. In the case of clustering, this interpretation is not valid. Assume two clusters and two

[1] *Probabilistic* implies, that the positive assignments of a data point to the clusters or classes sums up to 1. For *possibilistic* methods this restriction is dropped.

Figure 1.1: *A situation in which the probabilistic assignment of membership is counterintuitive for data sample X_2.*

isolated data samples as depicted in Fig. 1.1. Intuitively, one would state that the probability of data sample 1 to belong to one of the clusters is higher than the probability that data sample 2 belongs to them. The nature of clustering, being an ill-posed problem, suggests that each data sample has to be assigned to a cluster no matter how far away it is located. According to KRUSE ET AL. (Kruse et al. 2007) a better formulation is: *If a data sample has to be assigned to a cluster, then with the probability P to cluster C.*

A special challenge is to evaluate the obtained fuzzy clusterings and it is necessary to develop appropriate validation measures along with the algorithms. Therefore, a modification of the Fleiss' Kappa Value, which is applicable for fuzzy solutions, is presented. Although intended for the verification of classifications, it can also be applied to fuzzy cluster solutions.

Another aspect covered within this study is *relevance learning* versus *relevance clustering*. *Relevance learning* can be incorporated into classification algorithms to improve the separation of the classes. It results in the identification of less relevant vector components, which can be necglected to reduce the number of input dimensions (Hammer and Villmann 2002). *Relevance clustering* refers to a weighting of the input dimensions to enhance the cluster solution in terms of separation and compactness.

Further, in articles concerning the above mentioned prototype based methods often a phrase like *"the Euclidean distance is used but any other dissimilarity measure can be applied as well"* can be found. In this thesis some examples of non-standard metrics have been explored. For example section 3.1 demonstrates the usefulness and applicability of generalized divergences to cluster functional data. To cluster

non-vectorial data like text documents a median variant of FCM is introduced in section 3.2. In the there presented example the Kolmogorov complexity (Bennett et al. 1998, Vitányi and Li 2000), see also section 2.3.4, is applied to obtain the dissimilarities beforehand. The algorithm itself works with the provided dissimilarity matrix.

1.2 Outline

In the following chapter *"2 General notes about clustering and classification"* some of the most important prototype based cluster algorithms and classification methods are introduced. Among them are the well-known c-Means (CM), Neural Gas (NG), and different variants of Learning Vector Quantization (LVQ). The purpose of this chapter is to establish a comprehensive foundation including algorithms, cost functions, and syntax and notation conventions concerning those approaches, which will be relevant in the following chapters. Further, alternative distance measures as well as measures for the validation of clustering and classification results are presented. Section 2.4.3 contains a proposal of an extension of the Fleiss' Kappa Index, which allows the evaluation of fuzzy clusterings and classifications.

In chapter *"3 Fuzzy clustering for non-standard metrics"* fuzzy variants of the c-Means algorithm, e. g. Median Fuzzy c-Means (M-FCM) and Relevance Fuzzy c-Means (R-FCM), and a fuzzy interpretation of Affinity Propagation (AP) are proposed. The main property of M-FCM and FAP is, that both are based on mere similarities respectively dissimilarities between the data samples. No further knowledge about special features of the data points themselves is required. Therefore, these cluster algorithms are applicable for non-metric data. The R-FCM algorithm in turn is designed to adapt the influence of specific data dimensions by manipulating the metric properties to improve the clustering in terms of separation and compactness. Further it is demonstrated that the usually used Euclidean metric can be substituted by some other dissimilarity measure like a divergence.

Chapter *"4 Fuzzy Classification"* first introduces fuzzy variants of the Robust Soft LVQ and Soft NCP. Thereby, the class labels are assumed to be probabilistic assignments to the classes. Beside the derivation of the update rules, for both algorithms representative examples are provided. Further two semi-supervised fuzzy learning algorithms are proposed: Fuzzy Labeled SOM (FLSOM) and Fuzzy Labeled NG (FLNG). For these two also the theoretical aspects of the concept of relevance learning are presented.

Finally, the last chapter provides a summary and conclusions, points out open questions, and brings up ideas for future work.

Parts of chapter based on:

Dietlind Zühlke, Tina Geweniger, Ulrich Heimann, and Thomas Villmann – *"Fuzzy Fleiss-Kappa for comparison of fuzzy classifiers,"* in M. Verleysen (ed.), Proc. Of European Symposium on Artificial Neural Networks (ESANN 2009), d-side publications, Evere, Belgium, pp. 269-274, 2009.

Chapter 2

General notes about clustering and classification

Abstract

This chapter provides a general introduction into the field of clustering and classification. The basic concepts of unsupervised and supervised learning algorithms relevant in the following chapters are presented along with annotations concerning used symbols, syntax, and notation conventions. Furthermore, a number of suitable distances respectively dissimilarities and validation measures are specified.

This chapter covers the established methods and measures essential for the following chapters. The first two sections 2.1 and 2.2 refer to prototype based clustering and classification methods. Thereafter, different metric and non-metric distance measures are introduced in section 2.3, and finally, section 2.4 is concerned with a variety of validation measures applicable to fuzzy clustering and classification.

2.1 Vector Quantization and Clustering

Vector Quantization (VQ) in general is a prototype based learning scheme which can be applied for the sparse representation of unlabeled data. It is also referred to as unsupervised learning and used for the compression of very large data sets. Thereby, each cluster or group of similar objects is represented by one or more prototypes. These prototypes are located within the input space of the data samples. The main task of VQ algorithms is to describe the underlying data as close as possible. Respective information about the relationship between data points or between data points and prototypes are taken into account, for example dissimilarities or neighboorhood rankings.

In the following, the first section provides the basic concepts of VQ, and afterwards some well known cluster algorithms are described in more detail. The last

two methods – Median c-Means and Affinity Propagation – work with non-vectorial data and, therefore, are no vector quantizers in this sense. Yet, since they are also intended to cluster large data sets they are included into this chapter.

2.1.1 Vector Quantization

As mentioned before, VQ is among others a method used for data compression. Possibly high-dimensional data $\boldsymbol{v}_i \in \mathbb{R}^d$ given by the dataset $V = \{\boldsymbol{v}_1, ..., \boldsymbol{v}_{N_V}\}$ are mapped to a finite index set $A = \{1, \ldots, N_P\}$, which is associated with a set W of N_P codebook vectors (prototypes) $\boldsymbol{w}_j \in \mathbb{R}^d$. The codebook vectors are also called reference vectors. The set W respectively A is substantially smaller than V, $N_P < N_V$, and each data sample $\boldsymbol{v}_i$ is represented by the closest codebook vector $\boldsymbol{w}_{s(\boldsymbol{v}_i)}$. Formally this mapping can be stated as

$$\Psi_{V \to A} : \boldsymbol{v}_i \longmapsto s = \underset{j \in A}{\operatorname{argmin}}(d(\boldsymbol{v}_i, \boldsymbol{w}_j)) \tag{2.1}$$

where $d(\boldsymbol{v}_i, \boldsymbol{w}_j)$ usually is the squared Euclidean norm [1]

$$\begin{aligned} d(\boldsymbol{v}_i, \boldsymbol{w}_j) &= \| \boldsymbol{v}_i - \boldsymbol{w}_j \|^2 \\ &= (\boldsymbol{v} - \boldsymbol{w})^2. \end{aligned} \tag{2.2}$$

The mapping rule $\Psi_{V \to A}$ (2.1) realizes a *winner takes all* rule, where each codebook vector $\boldsymbol{w}_j$ represents a voronoi cell. All data samples within this receptive field

$$R_j = \{\boldsymbol{v}_i \mid \Psi_{V \to A}(\boldsymbol{v}_i) = j\} \tag{2.3}$$

are mapped to the respective codebook vector $\boldsymbol{w}_j$.

The crucial point in Vector Quantization is to find optimal codebook vectors to represent the data as accurate as possible. One way is to minimize the average expected square of the quantization error

$$E_{VQ} = \int d(\boldsymbol{v}, \boldsymbol{w}_{s(\boldsymbol{v})}) P(\boldsymbol{v}) d\boldsymbol{v} \tag{2.4}$$

where $P(\boldsymbol{v})$ is the continuous propability density of the data distribution within the input space V. For discrete data the integral is reduced to a sum over all samples. A simple method to minimize (2.4) is to reposition the codebook vectors following a

[1] Throughout this work, the term $d(\boldsymbol{v}, \boldsymbol{w})$ is used as a general distance or dissimilarity measure. Although most algorithms presented in this section are originally based on the squared Euclidean distance $d(\boldsymbol{v}, \boldsymbol{w}) = (\boldsymbol{v} - \boldsymbol{w})^2 = \sum_k (v_k - w_k)^2$, this is considered as a special case only and indicated seperately where appropriate. If the general term *distance* is used, a (dis)similarity measure rather than a mathematical distance is required.

stochastic gradient descent as described by KUSHNER&CLARK (Kushner and Clark 1978). This method was applied by LINDE, BUZO and GRAY, who developed the LBG-algorithm 2.1.1a (Linde et al. 1980). Starting with (random) initial values $\boldsymbol{w}_j(0)$ and updating $\boldsymbol{w}_j$ based on prior values of $\boldsymbol{w}_j$ according to

$$\boldsymbol{w}_j(t+1) = \boldsymbol{w}_j(t) + \Delta\boldsymbol{w}_j \tag{2.5}$$

the algorithm converges for small learning rates $0 < \varepsilon \ll 1$ to a local minimum (Linde et al. 1979). The parameter t indicates successive time steps and $\Delta\boldsymbol{w}_j$ is obtained according to

$$\begin{aligned} \Delta\boldsymbol{w}_j &= -\frac{\varepsilon}{2} \cdot \frac{\partial E_{VQ}(W)}{\partial \boldsymbol{w}_j} \\ &= -\frac{\varepsilon}{2} \cdot \frac{\partial}{\partial \boldsymbol{w}_j} \left(\int d(\boldsymbol{v}, \boldsymbol{w}_{s(\boldsymbol{v})}) P(\boldsymbol{v}) d\boldsymbol{v} \right) \end{aligned} \tag{2.6}$$

which leads as a stochastic gradient to

$$\Delta\boldsymbol{w}_j = -\varepsilon \cdot \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_{s(\boldsymbol{v}_i)})}{\partial \boldsymbol{w}_j} \tag{2.7}$$

for given $\boldsymbol{v}_i$. If it is not indicated otherwise, the formula $\frac{\partial E_{VQ}}{\partial \boldsymbol{w}_j}$ is an abbreviation for this stochastic gradient.

Algorithm 2.1.1a – Standard Vector Quantization (VQ) „LBG–Algorithm"

1. Initialize the codebook vectors $\boldsymbol{w}_j \in W$ randomly
2. Determine all updates $\Delta\boldsymbol{w}_j$ according to eq. (2.7)

   $$\Delta\boldsymbol{w}_j = -\varepsilon \cdot \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_{s(\boldsymbol{v}_i)})}{\partial \boldsymbol{w}_j}$$

3. Update all prototypes $\boldsymbol{w}_j$ according to eq. (2.5)

   $$\boldsymbol{w}_j(t+1) = \boldsymbol{w}_j(t) + \Delta\boldsymbol{w}_j$$

4. Repeat steps 2. – 4. until convergence or manual stop

Since generally the data distribution $P(\boldsymbol{v}_i)$ is not known, equation(2.5) can be reformulated as

$$\boldsymbol{w}_{s(\boldsymbol{v}_i)}(t+1) = \boldsymbol{w}_{s(\boldsymbol{v}_i)}(t) + \Delta\boldsymbol{w}_{s(\boldsymbol{v}_i)} \tag{2.8}$$

with

$$\Delta \boldsymbol{w}_{s(\boldsymbol{v}_i)} = -\varepsilon \cdot \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_{s(\boldsymbol{v}_i)}(t))}{\partial \boldsymbol{w}_{s(\boldsymbol{v}_i)}(t)} \tag{2.9}$$

which is independent of $P(\boldsymbol{v}_i)$ and leads to an approximation of the integration (Linde et al. 1980). The complete algorithm is given in Alg. 2.1.1b. The respective update rule based on the Euclidean norm (2.2) yields

$$\Delta \boldsymbol{w}_{s(\boldsymbol{v}_i)} = \varepsilon \cdot (\boldsymbol{v}_i - \boldsymbol{w}_{s(\boldsymbol{v}_i)}). \tag{2.10}$$

Algorithm 2.1.1b – Standard Vector Quantization (VQ)

1. Initialize the codebook vectors $\boldsymbol{w}_j \in W$ randomly
2. Present an input vector $\boldsymbol{v}_i$
3. Determine the *winner* respectively the best matching unit $\boldsymbol{w}_{s(\boldsymbol{v}_i)}$ according to eq. (2.1)

   $$s(\boldsymbol{v}_i) = \underset{j' \in A}{\operatorname{argmin}} \parallel \boldsymbol{v}_i - \boldsymbol{w}_{j'} \parallel$$

4. Update the *winner* $\boldsymbol{w}_{s(\boldsymbol{v}_i)}$ according to eq. (2.9)

   $$\boldsymbol{w}_{s(\boldsymbol{v}_i)}(t+1) = \boldsymbol{w}_{s(\boldsymbol{v}_i)}(t) - \varepsilon \cdot \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_{s(\boldsymbol{v}_i)}(t))}{\partial \boldsymbol{w}_{s(\boldsymbol{v}_i)}(t)}$$

5. Repeat steps 2. – 5. until convergence or manual stop

After the presentation of all data points the codebook vectors $\boldsymbol{w}_j$ are positioned in the center of gravity of their respective receptive field. Problems in the stability of the behaviour of the algorithm are known to arise. These are caused by discontinous jumps of the index $s(\boldsymbol{v}_i)$, which might occur during the update of the codebook vectors.

2.1.2 c-Means

The c-Means (CM) algorithm (also referred to as k-Means) is a simple and intuitive clustering method. It is a vector quantizer and as the name implies, this algorithm

groups a dataset into c distinct clusters, where each cluster is represented by a prototype $\boldsymbol{w}_j \in \mathbb{R}^d$ located at the mean, i. e. the center of gravity, of the respective cluster. There exists a batch as well as an online version.

Online c-Means For the case that the data distribution is not known a priori MACQUEEN proposed an online version of the c-Means algorithm (MacQueen 1967), where the data points are presented one after the other. At each time step a data point $\boldsymbol{v}_i$ is assigned to the closest prototype $\boldsymbol{w}_j$ based on a distance function $d(\boldsymbol{v}_i, \boldsymbol{w}_j)$ and only this prototype $\boldsymbol{w}_j$ is repositioned to the mean of the cluster. The method is straightforward and optimizes the general cost function given in equation (2.4) by stochastic gradient descent. The online update of all cluster centers $\boldsymbol{w}_j$ is determined by

$$\Delta \boldsymbol{w}_j = \varepsilon \cdot \delta_{j,s(\boldsymbol{v}_i)} \cdot \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j} \tag{2.11}$$

where ε is the step size and $\delta_{j,s(\boldsymbol{v}_i)}$ the Kronecker delta, which equals 1 iff $\boldsymbol{w}_j$ is the best matching prototype. For the special, originally proposed case, where the squared Euclidean distance is applied, this update rule can be formulated as

$$\Delta \boldsymbol{w}_j = -\varepsilon \cdot \delta_{j,s(\boldsymbol{v}_i)} \cdot (\boldsymbol{v}_i - \boldsymbol{w}_j) \tag{2.12}$$

Batch c-Means If the data distribution $P(V)$ is known a priori, the batch version proposed by BALL & HALL (Ball and Hall 1967, Ball and Hall 1965) can be used. The (heuristic) objective function to be minimized yields

$$E_{CM} = \sum_{i=1}^{N_V} \sum_{j=1}^{N_P} u_j(\boldsymbol{v}_i) \cdot d(\boldsymbol{v}_i, \boldsymbol{w}_j) \tag{2.13}$$

The set $U = \{u_j(\boldsymbol{v}_i)\}$, where $u_j(\boldsymbol{v}_i)$ is used as an abbreviation for $u(\boldsymbol{v}_i, \boldsymbol{w}_j)$, defines the crisp membership of data point $\boldsymbol{v}_i \in \mathbb{R}^d$ to cluster $\boldsymbol{w}_j$ and is bound to the restrictions

$$u_j(\boldsymbol{v}_i) \in \{0, 1\} \quad \text{and} \quad \sum_{j=1}^{N_P} u_j(\boldsymbol{v}_i) = 1. \tag{2.14}$$

As before, $d(\boldsymbol{v}_i, \boldsymbol{w}_j)$ denotes the distance between data point $\boldsymbol{v}_i$ and prototype $\boldsymbol{w}_j$. In the original proposal by BALL & HALL (Ball and Hall 1965) the squared Euclidean distance was used, but other non-negative dissimilarity measures might be suitable as well. A selection of different distance measures is presented in section 2.3.

The cost function (2.13) is optimized following an *alternating optimization* scheme:

Assignment update with fixed prototypes $\boldsymbol{w}_j$

$$u_j(\boldsymbol{v}_i) = \begin{cases} 1 & \text{if } d(\boldsymbol{v}_i, \boldsymbol{w}_j) = \min\{d(\boldsymbol{v}_i, \boldsymbol{w}_1), \ldots, d(\boldsymbol{v}_i, \boldsymbol{w}_{N_P})\} \\ 0 & \text{otherwise} \end{cases} \tag{2.15}$$

Prototype update with fixed assignments $u_j(\boldsymbol{v}_i)$ and based on the squared Euclidean distance

$$\boldsymbol{w}_j = \frac{\sum_{i=1}^{N_V} u_j(\boldsymbol{v}_i) \cdot \boldsymbol{v}_i}{\sum_{i=1}^{N_V} u_j(\boldsymbol{v}_i)} \tag{2.16}$$

Algorithm 2.1.2 – c-Means (CM)

1. Initialize the protoypes $\boldsymbol{w}_j \in W$ randomly
2. Calculate the memberships $u_j(\boldsymbol{v}_i)$ of all datapoints $\boldsymbol{v}_i$ to all prototpyes $\boldsymbol{w}_j$ according to eq. (2.15)

$$u_j(\boldsymbol{v}_i) = \begin{cases} 1 & \text{if } d(\boldsymbol{v}_i, \boldsymbol{w}_j) = \min\{d(\boldsymbol{v}_i, \boldsymbol{w}_1), \ldots, d(\boldsymbol{v}_i, \boldsymbol{w}_{N_P})\} \\ 0 & \text{otherwise} \end{cases}$$

3. Update the prototypes $\boldsymbol{w}_j$ according to (2.16)

$$\boldsymbol{w}_j = \frac{\sum_{i=1}^{N_V} u_j(\boldsymbol{v}_i) \cdot \boldsymbol{v}_i}{\sum_{i=1}^{N_V} u_j(\boldsymbol{v}_i)}$$

4. Repeat steps 2. – 4. until convergence or manual stop

Although both version of the c-Means algorithm always converge, they might not find the optimal clustering, since they tend to get stuck in local minima (Duda and Hart 1973). Therefore it is necessary to perform several runs with different prototype initializations. To select the initial prototypes several methods have been proposed. One of these suggests to take an appropriate number of data points as initial prototypes, or another method recommends to pick random points from the smallest (hyper-)box that encloses all data (Simpson 1993). A more sophisticated initialization method based on Latin hypercube sampling has been proposed by McKay (McKay et al. 1979).

2.1.3 Self-Organizing Maps

Self Organizing Maps (SOM) are also known as Kohonen Maps since they where introduced by KOHONEN (Kohonen 1990). They belong to the most popular data mining and visualization methods. Possibly high-dimensional data samples are nonlinearly mapped to a low-dimensional, typically two-dimensional grid. There, the topology of the original data is preserved. This grid or map constitutes a discrete representation of data samples $\boldsymbol{v}_i \in \mathbb{R}^d, i = 1, ..., N_V$. The nodes of the map act as the prototypes $\boldsymbol{w}_j \in \mathbb{R}^d$ and their position within the usually rectangular or *hexagonal* (actually triangular) grid is fixed. The training itself is administered by an unsupervised learning scheme based on vector quantization und uses a neighborhood function to preserve the topological properties of the input space.

The original model proposed by KOHONEN was improved by HESKES (Heskes 1999). According to (Heskes 1999) this modified method usually leads to almost the same results as the original SOM, yet additionally a cost function can be established, which makes this model feasible for continuous data distributions $P(V)$. In the following only HESKES' model is described.

A SOM consists of a set A of N_P neurons j, which are equipped with weight vectors $\boldsymbol{w}_j$ representing the prototypes. The neurons are arranged on a lattice structure, which determines the neighborhood relation $N(j, l)$ of neuron j and l. The mapping description of a trained SOM is defined by

$$\Psi_{\boldsymbol{v}\to A} : \boldsymbol{v} \longmapsto s(\boldsymbol{v}) = \underset{j\in A}{\operatorname{argmin}} \sum_{l\in A} h_\sigma(j, l) d(\boldsymbol{v}, \boldsymbol{w}_l) \tag{2.17}$$

where

$$h_\sigma(j, l) = \exp\left(-\frac{N(j, l)}{\sigma^2}\right) \tag{2.18}$$

describes the neighborhood cooperation with range $\sigma > 0$. The dissimilarity specifying $d(\boldsymbol{v}_i, \boldsymbol{w}_j)$ is an appropriate differentiable distance measure, usually the standard Euclidean norm. In this method, an input is mapped onto that position $j = s(\boldsymbol{v}_i)$ of the SOM, where the distance $d(\boldsymbol{v}_i, \boldsymbol{w}_j)$ is at a minimum. Thereby the average over all neurons according to the neighborhood is taken into account. The neuron $s(\boldsymbol{v}_i)$ is referred to as the *winner*.

A randomly initialized SOM is trained by sequential stochastic presentation of data points $\boldsymbol{v}_i \in V$. At each time step the closest neuron $s(\boldsymbol{v}_i)$ according to (2.17) is determined and the weights of neighboring neurons are adapted by

$$\Delta \boldsymbol{w}_j = -\varepsilon h_\sigma(j, s(\boldsymbol{v}_i)) \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j} \tag{2.19}$$

with a learning rate $\varepsilon > 0$ and $\varepsilon \to 0$. This adaptation follows a stochastic gradient descent of the cost function introduced by HESKES (Heskes 1999):

$$E_{SOM} = \frac{1}{C(\sigma)} \int P(V) \sum_{j=1}^{N_P} \delta_{j,s(\boldsymbol{v}_i)} \sum_{l=1}^{N_P} h_\sigma(j,l) \cdot d(\boldsymbol{v}_i, \boldsymbol{w}_l) d\boldsymbol{v}_i \tag{2.20}$$

where σ is decreased with successive time steps, and $\delta_{j,s(\boldsymbol{v}_i)}$ is the Kronecker symbol checking the identity of j and $s(\boldsymbol{v}_i)$.

Algorithm 2.1.3 – Self-Organizing Map (SOM) HESKES

1. Initialize the neurons weights $\boldsymbol{w}_j$ randomly
2. Present a data point $\boldsymbol{v}_i$ and identify the *winner* neuron (best matching unit) $s(\boldsymbol{v}_i)$ according to eq. (2.17)

   $$s(\boldsymbol{v}_i) = \underset{j \in A}{\operatorname{argmin}} \sum_{l \in A} h_\sigma(j,l) d(\boldsymbol{v}_i, \boldsymbol{w}_l)$$

3. Update the weights of the neurons in the neighborhood of the *winner* $s(\boldsymbol{v}_i)$ according to eq. (2.19), i. e. pull them closer to the input vector

   $$\boldsymbol{w}_j(t+1) = \boldsymbol{w}_j(t) - \varepsilon h_\sigma(j, s(\boldsymbol{v}_i)) \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j(t))}{\partial \boldsymbol{w}_j(t)}$$

4. Repeat steps 2. – 4. until convergence or manual stop

One main aspect of SOMs is the visualization ability of the resulting map due to its topological structure (Villmann et al. 1997). Under certain conditions the resulting non-linear projection $\Psi_{v \to A}$ generates a continuous mapping from the data space V onto the grid structure A. This mapping can mathematically be interpreted as an approximation of the principal curve or its higher-dimensional equivalents (Hastie and Stuetzle 1989). Thus, similar data points are projected on prototypes which are neighbored in the grid space A. Further, prototypes neighbored in the lattice space code similar data properties, i. e. their weight vectors are close together in the data space according to the applied metric. This property of SOMs is called topology preserving (or topographic) mapping realizing the mathematical concept of continuity (Villmann et al. 1997).

2.1.4 Neural Gas

The Neural Gas (NG) algorithm by MARTINETZ, BERKOVICH & SCHULTEN (Martinetz et al. 1993) is a highly efficient clustering algorithm. As for the algorithms in the previous sections the goal is to find prototypes $\boldsymbol{w}_j \in \mathbb{R}^d, j = 1, \ldots, N_P$ to represent continuous data points $\boldsymbol{v} \in \mathbb{R}^d$ which are distributed according to an underlying distribution as accurate as possible. Thereby, as known for SOMs, the neighborhood of the prototypes is taken into account, yet without restricting the prototypes to fixed grid positions. Instead, a rank function based on the distances between the prototypes $\boldsymbol{w}_i$ and data points $\boldsymbol{v}$ is introduced. Therefore, the dynamics of the prototypes during the adaption process resemble the dynamics of Brownian particles moving in a potential determined by the data density (Martinetz et al. 1993). A major advantage of this algorithm compared to c-Means is, that the NG is insensitive to the prototype initialization.

The cost function to minimize is given by

$$E_{NG} = \frac{1}{C(\sigma)} \sum_{j=1}^{N_P} \int h_\sigma(k_j(\boldsymbol{v}, W)) \cdot d(\boldsymbol{v}, \boldsymbol{w}_j) P(V) d\boldsymbol{v}. \tag{2.21}$$

In the original proposal the Euclidean distance was used for $d(\boldsymbol{v}, \boldsymbol{w})$. The function

$$k_j(\boldsymbol{v}, W) = |\{\boldsymbol{w}_k | d(\boldsymbol{v}, \boldsymbol{w}_k) < d(\boldsymbol{v}, \boldsymbol{w}_j)\}| \tag{2.22}$$

denotes the rank of the prototypes sorted according to their distances to the respective data point $\boldsymbol{v}$. The neighborhood function $h_\sigma(t)$ returns a maximum value for $k_j(\boldsymbol{v}, W) = 0$ and decreases to zero for higher ranks. Commonly, a Gaussian shaped curve $h_\sigma(t) = \exp(-t/\sigma)$ with neighborhood range $\sigma > 0$ is chosen. The normalization constant $C(\sigma) = \sum_{j=1}^{N_P} h_\sigma(k_j)$ depends only on σ.

The learning is accomplished by stochastic gradient descend with respect to $\boldsymbol{w}_j$. Given a data point $\boldsymbol{v}$ the update rule for prototype $\boldsymbol{w}_j$ yields

$$\Delta \boldsymbol{w}_j = -\varepsilon \cdot h_\sigma(k_j(\boldsymbol{v}, W)) \cdot \frac{\partial d(\boldsymbol{v}, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j} \tag{2.23}$$

where ε is the learning rate with $\varepsilon > 0$. The neighborhood range σ is decreased during the training, which ensures independence of the initialization at the beginning of the training and optimizes the quantization error towards the end. For $\sigma \to 0$ the update rule is equivalent to the c-Means update rule eq. (2.11), and for $\sigma \neq 0$ all prototypes within the range of σ are proportionally updated, i. e. not just the winner, but also the second, third, etc. ranked prototype (Martinetz et al. 1993).

Since its proposal the NG algorithm has been subject to extensive research and a number of variants have been developed. Prominent extensions are the Growing

Algorithm 2.1.4 – Neural Gas (NG)

1. Initialize the protoypes $\boldsymbol{w}_j \in W$ randomly
2. Present a randomly chosen data point $\boldsymbol{v}$
3. Update all prototypes according to eq. (2.23)

$$\boldsymbol{w}_j(t+1) = \boldsymbol{w}_j(t) + \varepsilon \cdot h_\sigma(k_j(\boldsymbol{v}, W)) \cdot \frac{\partial d(\boldsymbol{v}, \boldsymbol{w}_j(t))}{\partial \boldsymbol{w}_j(t)}$$

4. Repeat steps 2. – 4. until convergence or manual stop

Neural Gas (Fritzke 1995), the Supervised Neural Gas (Villmann and Hammer 2002, Hammer et al. 2005), the Kernel Neural Gas (Qin and Suganthan 2004a), and the Batch Neural Gas (Cottrell et al. 2006). Since the batch variant is of interest in section 4.2.1, a short review of the algorithm will be presented in the following.

According to COTTRELL ET AL. (Cottrell et al. 2006) the Neural Gas cost function for discrete data $\boldsymbol{v}_i \in \mathbb{R}^d, i = 1, \ldots, N_V$ can be approximated by

$$E_{NG\_Batch} = \frac{1}{C(\sigma)} \sum_{j=1}^{N_P} \sum_{i=1}^{N_V} h_\sigma(k_j(\boldsymbol{v}_i, W)) \cdot d(\boldsymbol{v}_i, \boldsymbol{w}_j). \tag{2.24}$$

The quantities $k_{ij} := k_j(\boldsymbol{v}_i, W)$ are now treated as hidden variables with the constraint, that the values $k_{ij}(j = i, ..., N_P)$ constitute a permutation of $\{0, \ldots, n-1\}$ for each data point $\boldsymbol{v}_i$. Since the cost function depends on the hidden variables k_{ij} and the prototypes $\boldsymbol{w}_j$, it has to be optimized with respect to k_{ij} and $\boldsymbol{w}_j$ yielding alternating adaptation steps:

Determine the ranks of the prototypes $\boldsymbol{w}_j$

$$k_{ij} = |\{\boldsymbol{w}_l | d(\boldsymbol{v}_i, \boldsymbol{w}_l) < d(\boldsymbol{v}_i, \boldsymbol{w}_j)\}| \tag{2.25}$$

Update the prototypes based on fixed ranks k_{ij} according to eq. (2.23)
Using the squared Euclidean distance this update can be formulated as

$$\boldsymbol{w}_j = \frac{\sum_{i=1}^{N_V} h_\sigma(k_{ij}) \cdot \boldsymbol{v}_i}{\sum_{i=1}^{N_V} h_\sigma(k_{ij})} \tag{2.26}$$

As known from other batch algorithms, before the update can take place all data samples have to be presented. Usually only a few, i. e. 10 to 100, adaptation steps are necessary.

2.1.5 Fuzzy c-Means

The Fuzzy c-Means (FCM) is an extension of the standard c-Means algorithm and was proposed by DUNN (Dunn 1973) and further on extensively discussed and improved by BEZDEK (Bezdek 1980), HATHAWAY (Hathaway and Bezdek 1986, Hathaway et al. 1989) and others (Cannon et al. 1986, Ismail and Selims 1986). It is also an unsupervised learning scheme which aims to minimize an objective function by alternating update steps for prototypes and cluster memberships. The membership assigning data point $\boldsymbol{v}_i$ to prototype $\boldsymbol{w}_j$ now is no longer crisp, i. e. a data point can be assigned to one or more clusters. This assignment can either be possibilistic or probabilistic depending on the contraints put on $u_j(\boldsymbol{v}_i)$:

- probabilistic: $u_j(\boldsymbol{v}_i) \geq 0 \quad$ with $\quad \sum_{j=1}^{N_P} u_j(\boldsymbol{v}_i) = 1, \quad \forall i \in \{1, \ldots, N_V\}$
- possibilistic: $u_j(\boldsymbol{v}_i) \in [0, 1]$

Throughout this work only the probabilistic version will be persued. Therefore the $N_V \times N_P$ matrix U with the elements $u_j(\boldsymbol{v}_i)$ now is a probabilistic cluster partition of V. The respective objective function to minimize is given by

$$E_{FCM} = \sum_{i=1}^{N_V} \sum_{j=1}^{N_P} u_j(\boldsymbol{v}_i)^m \cdot d(\boldsymbol{v}_i, \boldsymbol{w}_j) \tag{2.27}$$

The exponent $m \in (1, \infty)$ regulates the *fuzziness*. For $0 \leftarrow m$ the assignments converge to crisp decisions, wheras $m \to \infty$ forces equally distributed assignments. Usually the *fuzziness* is set to $1.2 \leq m \leq 2$. (Bezdek 1980)

Analogously to the standard c-Means the updates for the assignments $u_j(\boldsymbol{v}_i)$ and the prototypes $\boldsymbol{w}_j$ follow an alternating optimization scheme, yet also take the *fuzziness* m into account:

Assignment update with fixed prototypes $\boldsymbol{w}_j$

$$\begin{aligned} u_j(\boldsymbol{v}_i) &= \frac{1}{\sum_{k=1}^{N_P} \left(\frac{d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{d(\boldsymbol{v}_i, \boldsymbol{w}_k)} \right)^{\frac{1}{m-1}}} \\ &= \frac{\sqrt[m-1]{d(\boldsymbol{v}_i, \boldsymbol{w}_j)^{-1}}}{\sum_{k=1}^{N_P} \sqrt[m-1]{d(\boldsymbol{v}_i, \boldsymbol{w}_k)^{-1}}} \end{aligned} \tag{2.28}$$

Prototype update with fixed assignments $u_j(\boldsymbol{v}_i)$

$$\boldsymbol{w}_j = \frac{\sum_{i=1}^{N_V} u_j(\boldsymbol{v}_i)^m \cdot \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j}}{\sum_{i=1}^{N_V} u_j(\boldsymbol{v}_i)^m} \tag{2.29}$$

The update rules (2.28) and (2.29) are derived as solutions of the Lagrange minimization problem

$$\begin{aligned} L_{FCM}(\boldsymbol{\lambda}) &= E_{FCM} + \sum_{i=1}^{N_V} \lambda_i \left(\sum_{j=1}^{N_P} u_j(\boldsymbol{v}_i) - 1 \right) \\ &= \frac{1}{2} \sum_{j=1}^{N_P} \sum_{i=1}^{N_V} u_j(\boldsymbol{v}_i)^m \cdot d(\boldsymbol{v}_i, \boldsymbol{w}_j) + \sum_{i=1}^{N_V} \lambda_i \left(\sum_{j=1}^{N_P} u_j(\boldsymbol{v}_i) - 1 \right) \end{aligned} \quad (2.30)$$

For the special case of the quadratic Euclidean distance (2.2) eq. (2.29) yields

$$\boldsymbol{w}_j = \frac{\sum_{i=1}^{N_V} u_j(\boldsymbol{v}_i)^m \cdot \boldsymbol{v}_i}{\sum_{i=1}^{N_V} u_j(\boldsymbol{v}_i)^m} \quad (2.31)$$

2.1.6 Median c-Means

The Median c-Means (M-CM) is another variant of the classical c-Means (Bezdek 1981). Yet, in contrast to the classical c-Means, the requirement that the data samples have to be embedded in a metric or vector space has been dropped. It is merely necessary to have some dissimilarity defined between the data samples, whereby this dissimilarity should follow the common sense understanding of *dissimilarity* (Pękalska and Duin 2005). The clustering now restricts the prototypes to be data samples itself: the algorithm determines a set $W = \{w_1, \ldots, w_{N_P}\}$ where $W \subset V$. Instead of a distance function $d(v_i, w_j)$ now only a dissimilarity matrix D with $N_V \times N_V$ elements has to be provided. The elements are referred to as $d(v_i, v_k)$ and D is assumed to be complete and symmetric.

Let P be the index set $P = \{1, \ldots, N_P\}$ for the prototypes and $S = \{1, \ldots, N_V\}$ the index set for the data samples, then the mapping

$$I(i) : S \to P \quad (2.32)$$

specifies the winner index for a given data sample v_i. This winner index refers to the index of the prototype with minimum distance to v_i:

$$I(i) = \underset{j \in P}{\operatorname{argmin}}\, d(v_i, w_j) \quad (2.33)$$

Since the prototypes w_j are restricted to be data samples, the similarities $d(v_i, w_j)$ are given by the distance matrix D. Let $J(j)$ denote the respective data index defining a bijective mapping

$$J(j) : P \to S' \subset S \quad (2.34)$$

then the winner rule (2.33) can be written as

$$I(i) = \underset{j \in P}{\operatorname{argmin}}\, d(v_i, w_{J(j)}) \tag{2.35}$$

and thus the cost function for M-CM is given by

$$E_{M-CM} = \frac{1}{2} \sum_{i=1}^{N_V} \sum_{j=1}^{N_P} u_{i,I(i)} \cdot d(v_i, v_{J(j)}) \tag{2.36}$$

$$= \frac{1}{2} \sum_{i=1}^{N_V} d(v_i, v_{J(I(i))}) \tag{2.37}$$

where the characteristic function $u_{i,I(i)} = 1$ iff the data sample v_i is represented by prototype $w_{I(i)}$ which is equivalent with data sample $v_{J(I(i))}$.

Analogously to the in section 2.1.2 described c-Means algorithm, the optimization of the Median c-Means follows an alternating update of prototypes and assignments:

1. *Assignment update*: determine for each data sample $v_i \in V$ the winner $I(i) \in P$ according to the minimum rule (2.33)

2. *Prototype update*: choose new prototypes $w_j = v_l$ according to

$$l = \underset{l'}{\operatorname{argmin}} \sum_{i=1}^{N_V} u_{i,l'} \cdot d(v_i, v_{l'}). \tag{2.38}$$

Compared to the standard c-Means this median variant achives a considerable speed-up especially for large data set, since repeated calculations of distances between data samples and prototypes can be skipped. Instead, the required dissimilarities are obtained from the distance Matrix D which has to be provided.

The major drawback is the restriction to use data samples as prototypes. Especially dealing with disjunct clusters leads to an only suboptimal solution. The prototypes have to be initialized carefully to avoid getting stuck in local minima.

2.1.7 Affinity Propagation

Affinity propagation as proposed by FREY & DUECK (Frey and Dueck 2007) is another prototype based clustering algorithm, yet in contrast to the other algorithms it is based on *similarities* between data samples. It has been successfully applied for face recognition, the optimization of flight connections based on traveling time, to detect genes in microarray data, and to identify representative sentences in a manuscript (Frey and Dueck 2007).

The data points can be interpreted as nodes in a network and initially all of them are treated as prospective canditates for prototypes. By exchanging real-valued message simultaneously, the data points *communicate* with each other and gradually a number of them will be recognized as representative cluster centers, i. e. prototypes, and clusters will emerge. Thereby, the number of the prototypes is initially not known, but can indirectly be influenced by a certain parameter. The name *affinity propagation* originates in the nature of the exchanged messages, which express the affinity a data point has to chose another data point as its prototype. The actual values of the exchanged messages are based on simple formulas that search for minima of an appropriately chosen energie function based on factor graphs.

The algorithm takes real-valued similarities $s(i,k)$ of N_V data points $v_i, v_k \in N_V$ into account. In general these similarities do not have to be symmetric, i. e. $s(i,k) \neq s(k,i)$. The algorithm is also suited to deal with incomplete data in the sense that not all similarities have to be known. Since the algorithm is based on similarities only, it can also be used for clustering non-vectorial data. If dealing with vectorial data $\boldsymbol{v} \in \mathbb{R}^d$, any negative distance measure, e. g. the negative squared error (Euclidean distance) $s(i,k) = -(\boldsymbol{v}_i - \boldsymbol{v}_k)^2$, could be applied.

The initial value of $s(i,i)$ is the so called *preference*. The higher the initial preference of a data point the more likely this data point will be chosen as a prototype. To even the chances for all datapoints this value can also be set to a common value for all of them, e. g. the median or minimum of all similarities $s(i,k)$. The higher the initial values of all $s(i,i)$, the more clusters will eventually emerge (Frey and Dueck 2007).

There are two types of real-valued messages, which are in competition with each other and are exchanged between the data points simultaneously: *responsibilities* $r(i,k)$ and *availabilities* $a(i,k)$. While the first type, sent from data point v_i to data point v_k, indicates how well suited data point v_k would be as a prototype for v_i (taking all other potential prototypes into account), the avalabilities are sent the other direction from v_k to v_i and indicate how appropriate v_k would be as an prototype for v_i (this time taking other data points and their affinity for v_k into account). Both message types can be interpreted as log-probability ratios and are updated according to a two-step alternating iteration scheme. The responsibilities are adapted by keeping the availabilities stable according to

$$r(i,k) \leftarrow d(i,k) - \max_{l:\, l \neq k} \{a(i,l) + s(i,l)\} \tag{2.39}$$

in turn the availabilities are calculated based on fixed responsibilities

$$\begin{aligned} \forall\, i \neq k: \quad & a(i,k) \leftarrow \min\{0, r(k,k) + \sum_{l:l\neq\{i,k\}} \max\{0, r(l,k)\}\} \\ \forall\, i = k: \quad & a(k,k) \leftarrow \sum_{l:l\neq k} \max\{0, r(l,k)\} \end{aligned} \tag{2.40}$$

Since in both update rules the similarity between the two respective data points is considered, there are obviously no message exchanges between data points with unknown similarities.

The iterative alternating calculation of $a(i,k)$ and $r(i,k)$ is caused by the max-sum-algorithm applied for factor graphs (Pearl 1988), which can further be related to spectral clustering (Luxburg 2007).

And any point during the training the current prototype candidates can be identified by combining the active message values. The prototype candidate k for data point i is obtained according to

$$I(i) = \underset{l\in V_N}{\operatorname{argmax}}\{a(i,l) + r(i,l)\}. \tag{2.41}$$

Note that if $i = l$ the data point itself is a prototype.

Although not explicitly stated in (Frey and Dueck 2007), a cost function which has to be maximized can be formulated as

$$S(I) = \sum_i s\left(i, I(i)\right) + \sum_j \delta_j(I) \tag{2.42}$$

where $I : N \rightarrow N$ is the mapping function defining the prototypes for each data point. Thereby $\delta_j(I)$ is a penalty function

$$\delta_j(I) = \begin{cases} -\infty & \text{if } \exists j,k \quad I(j) \neq j, I(k) = j \\ 0 & \text{otherwise} \end{cases} \tag{2.43}$$

The algorithm, which follows a two step iteration scheme consisting of repeated message updates, stops if the change of the values of the exchange messages falls below a certain treshold or if a fixed number of predefined iterations is reached.

2.2 Prototype Based Classification

Nearest Prototype Classification (NPC) is a very simple classifier in pattern recognition, where an unlabeled data point is assigned to the class of the nearest prototype. Thereby, it is assumed that each class is represented by at least one appropriately chosen prototype which locally approximates the classification boundary. Since the

Algorithm 2.1.7 – Affinity Propagation (AP)

1. Inititialize all availabilities $a(i,k) = 0$
2. Calculate responsibilities according to eq. 2.39

$$r(i,k) \leftarrow d(i,k) - \max_{l:\ l \neq k} \{a(i,l) + s(i,l)\}$$

3. Calculate availabilities according to eq. 2.40

$$\forall\, i \neq k: \quad a(i,k) \leftarrow \min\{0, r(k,k) + \sum_{l: l \neq \{i,k\}} \max\{0, r(l,k)\}\}$$
$$\forall\, i = k: \quad a(k,k) \leftarrow \sum_{l: l \neq k} \max\{0, r(l,k)\}$$

4. Repeat steps 2.-4. until either
   - a fixed number of iterations is reached or
   - the value changes of the messages fall below a certain treshold or
   - manual stop
5. Identify the prototypes by combining the current message values according to eq. 2.41

$$I(i) = \arg\max_{l \in V_N} \{a(i,l) + r(i,l)\}$$

set of the prototypes is considerably smaller than the set of the data points, this method is computationally efficient. The classification of an unknown data point only requires comparisons with a few prototypes instead with all data points. For all NPCs, obviously the unknown data point is assigned to the closest prototype. The prototype training itself usually is based on Hebbian Learning, which provides a paradigm to obtain relatively fast and easy-to-use algorithms.

There are different methods to obtain these class representative prototypes. One large group are the Learning Vector Quantizers (LVQ), which are intuitive and well known for their stability solving a wide range of classification problems. The basic algorithm was proposed by KOHONEN in (Kohonen 1986), but meanwhile a number of variants have been developed. While the standard LVQ is based on heuristics aiming on the minimization of the classification error (Kohonen 1995), more ad-

vanced LVQ schemes like Generalized LVQ (GLVQ) (Sato and Yamada 1996), Kernel GLVQ (KGLVQ) (Qin and Suganthan 2004b), or Robust Soft LVQ (RSLVQ) (Seo and Obermayer 2003) replace the simple classification error by sophisticated cost functions which allow a gradient ascent/descent learning or optimization by expectation maximization.

Another method to create an NPC is the Soft Nearest Prototype Classifier (SNPC) proposed by SEO, BODE & OBERMAYER, which is based on a Gaussian mixture ansatz. An extension thereof is the fuzzy version by VILLMANN ET AL. (Villmann, Schleif and Hammer 2006), which allows fuzzy prototype assignments.

Generally, for all the approaches, the placement of the prototypes within a class depends on their distance to the respective data points. Assume labeled training data $\boldsymbol{v}_i \in \mathbb{R}^d, i = 1, ..., N_V$ with the class memberships $c(\boldsymbol{v}_i) \in C$ and prototypes $\boldsymbol{w}_j \in \mathbb{R}^d, j = 1, ..., N_P$ with $y(\boldsymbol{w}_j) \in C$ specifying the class assignments. Thereby, the finite index set $C \in \mathbb{N}$ with N_C elements identifies the respective classes and $1 \leq c(\boldsymbol{v}_i), y(\boldsymbol{w}_j) \leq N_C$. Based on an appropriate distance measure $d(\boldsymbol{v}_i, \boldsymbol{w}_j)$ in $\mathbb{R}^d$, the class prototypes are placed according to the class distributions, which itself are determined by the underlying metric. A common metric for the calculation of the similarity between prototypes and data points is the Euclidean distance[2]. But other dissimilarity measures might as well be suitable depending on the specific classification problem.

2.2.1 Learning Vector Quantization – LVQ1

This very basic supervised heuristic learning scheme was introduced by KOHONEN (Kohonen 1986) and aims to position the prototypes, also referred to as codebook vectors, to reach a high classification accuracy. Thereby, each data point and each prototype belong to only one class, but a class can be represented by more than one prototype.

During the training LVQ1 aims to minimize the classification error by repositioning the prototypes according to their class membership. If the prototype closest to the presented data point belongs to the same class as the data point itself, the prototype is moved closer to the data points. Otherwise, if the prototype belongs to a different class, it is pushed away. Formally, the update rule based on the nearest

[2]In this chapter the squared Euclidean distance $d(\boldsymbol{v}_i, \boldsymbol{w}_j) = (\boldsymbol{v}_i - \boldsymbol{w}_j)^2$ is used. Exceptions thereof are indicated explicitly.

neighbor is given by

$$\Delta \boldsymbol{w}_j = \begin{cases} \varepsilon \cdot (\boldsymbol{v}_i - \boldsymbol{w}_j)^2 & \text{if} \quad s(\boldsymbol{v}_i) = j \quad \text{and} \quad c(\boldsymbol{v}_i) = y(\boldsymbol{w}_j) \\ -\varepsilon \cdot (\boldsymbol{v}_i - \boldsymbol{w}_j)^2 & \text{if} \quad s(\boldsymbol{v}_i) = j \quad \text{and} \quad c(\boldsymbol{v}_i) \neq y(\boldsymbol{w}_j) \\ 0 & \text{if} \quad s(\boldsymbol{v}_i) \neq j \end{cases} \tag{2.44}$$

where ε is a small learning rate. Note that only the closest prototype is updated, i. e. only the *winner* is moved. According to KOHONEN this algorithm might show some instabilities near the class borders, since the number of wrongly classified data points might be higher than correctly classified samples and therefore the repulsion rule is applied more often. This difference may cause a small bias to the asymptotic values of the $\boldsymbol{w}_j$ (Kohonen 1995).

Algorithm 2.2.1 – Learning Vector Quantization LVQ1

1. Initialize labeled codebook vectors $\boldsymbol{w}_j \in W$ randomly
2. Present a labeled input vector $\boldsymbol{v}_i$
3. Determine the closest codebook vector $\boldsymbol{w}_{s(\boldsymbol{v}_i)}$ according to (2.45)

   $$s(\boldsymbol{v}_i) = \underset{j \in A}{\operatorname{argmin}} \, (\boldsymbol{v}_i - \boldsymbol{w}_j)^2$$

4. Update $\boldsymbol{w}_{s(\boldsymbol{v}_i)}$ according to equation (2.44)

   $$\boldsymbol{w}_{s(\boldsymbol{v}_i)}(t+1) = \boldsymbol{w}_{s(\boldsymbol{v}_i)}(t) + \begin{cases} -\varepsilon \cdot (\boldsymbol{v}_i - \boldsymbol{w}_{s(\boldsymbol{v}_i)}(t))^2 & \text{if} \quad c(\boldsymbol{v}_i) = y(\boldsymbol{w}_{s(\boldsymbol{v}_i)}) \\ \varepsilon \cdot (\boldsymbol{v}_i - \boldsymbol{w}_{s(\boldsymbol{v}_i)}(t))^2 & \text{if} \quad c(\boldsymbol{v}_i) \neq y(\boldsymbol{w}_{s(\boldsymbol{v}_i)}) \end{cases}$$

5. Repeat steps 2. – 5. until convergence or manual stop

An unlabeled data point $\boldsymbol{v}_i$ presented to a trained network is assigned to the class of the closest prototype respectively codebook vector $\boldsymbol{w}_j$, i. e. the best matching unit defined by

$$\boldsymbol{v}_i \longmapsto y(\boldsymbol{w}_{s(\boldsymbol{v}_i)}) \qquad s(\boldsymbol{v}_i) = \underset{j \in A}{\operatorname{argmin}} \, (\boldsymbol{v}_i - \boldsymbol{w}_j)^2 \tag{2.45}$$

where $s(\boldsymbol{v}_i)$ is the index of the *winner* and A the output space.

2.2.2 Learning Vector Quantization – LVQ2.1

LVQ2.1 is a modification of LVQ1. While the classification is still the same as in (2.45), i. e. assigning an unlabeled data sample to the class of the closest prototype, the adaption of the prototype positions during the training process is modified. Now the *two* closest prototypes of a presented training sample are updated simultanuously. Yet, an update step will be performed only if all of the following three conditions are fulfilled:

(i) one of the prototypes denoted as $\boldsymbol{w}_{j^+}$ belongs to the same class as the presented data sample $\boldsymbol{v}_i$: $c(\boldsymbol{v}_i) = y(\boldsymbol{w}_{j^+})$

(ii) the other prototype denoted as $\boldsymbol{w}_{j^-}$ belongs to a different class as the presented data sample $\boldsymbol{v}_i$: $c(\boldsymbol{v}_i) \neq y(\boldsymbol{w}_{j^-})$

(iii) the protoypes $\boldsymbol{w}_{j^+}$ and $\boldsymbol{w}_{j^-}$ are located within an active region specified by the *window* w

$$\min\left(\frac{(\boldsymbol{v}_i - \boldsymbol{w}_{j^+})^2}{(\boldsymbol{v}_i - \boldsymbol{w}_{j^-})^2}, \frac{(\boldsymbol{v}_i - \boldsymbol{w}_{j^-})^2}{(\boldsymbol{v}_i - \boldsymbol{w}_{j^+})^2}\right) > s \quad \text{, where} \quad s = \frac{1-w}{1+w} \tag{2.46}$$

The *window rule* had to be introduced to circumvent diverging prototypes. The update follows the rules of LVQ1, i. e. attracting the prototype $\boldsymbol{w}_{j^+}$ belonging to the same class as data sample $\boldsymbol{v}_i$ and repelling prototype $\boldsymbol{w}_{j^-}$ belonging to any other class:

$$\Delta\boldsymbol{w}_{j^+} = \varepsilon \cdot (\boldsymbol{v}_i - \boldsymbol{w}_{j^+})^2 \tag{2.47}$$

$$\Delta\boldsymbol{w}_{j^-} = -\varepsilon \cdot (\boldsymbol{v}_i - \boldsymbol{w}_{j^-})^2 \tag{2.48}$$

Again, ε is a small learning rate. Caution is necessary choosing the width of the window, since large differences in the repelling and attracting forces might lead to discontinuos behaviour during the optimization. The recommended value of w is between 0.2 and 0.3 (Kohonen 1995). The update causes the border between the receptive fields of the prototypes to be shifted.

2.2.3 Generalized LVQ

The LVQ algorithms mentioned in the previous chapters are based on simple but appropriate heuristics and aim to minimize the number of misclassifications, i. e. to optimize a discrete error function

$$E_{LVQ} = \sum_{i=1}^{N_V} (c(\boldsymbol{v}_i) - y(\boldsymbol{w}_{s(\boldsymbol{v}_i)})^2 \tag{2.49}$$

Algorithm 2.2.2 – Learning Vector Quantization LVQ2.1

1. Initialize the labeled codebook vectors $\boldsymbol{w}_j \in W$ randomly
2. Present a labeled input vector $\boldsymbol{v}_i$
3. Determine the *two* closest codebook vectors according to (2.45) respectively

$$s(\boldsymbol{v}_i) = \underset{j \in A}{\operatorname{argmin}} \, (\boldsymbol{v}_i - \boldsymbol{w}_j)^2$$

4. If all three conditions

   (i) $c(\boldsymbol{v}_i) = y(\boldsymbol{w}_{j^+})$

   (ii) $c(\boldsymbol{v}_i) \neq y(\boldsymbol{w}_{j^-})$

   (iii) $\min\left(\frac{(\boldsymbol{v}_i - \boldsymbol{w}_{j^+})^2}{(\boldsymbol{v}_i - \boldsymbol{w}_{j^-})^2}, \frac{(\boldsymbol{v}_i - \boldsymbol{w}_{j^-})^2}{(\boldsymbol{v}_i - \boldsymbol{w}_{j^+})^2}\right) > \frac{1-w}{1+w}$

   are fulfilled, update $\boldsymbol{w}_{j^+}$ and $\boldsymbol{w}_{j^-}$ according to (2.47) and (2.48)

$$\begin{aligned} \boldsymbol{w}_{j^+}(t+1) &= \boldsymbol{w}_{j^+}(t) - \varepsilon \cdot (\boldsymbol{v}_i - \boldsymbol{w}_{j^+}(t))^2 \\ \boldsymbol{w}_{j^-}(t+1) &= \boldsymbol{w}_{j^-}(t) + \varepsilon \cdot (\boldsymbol{v}_i - \boldsymbol{w}_{j^-}(t))^2 \end{aligned}$$

5. Repeat steps 2. – 5. until convergence or manual stop

which is not differentiable with respect to the prototypes $\boldsymbol{w}_j$. In 1995 SATO & YAMADA proposed a LVQ based learning scheme incorporating a *differentable* cost function approximating the classification error (Sato and Yamada 1996). This Generalized LVQ (GLVQ) allows stochastic gradient descent learning of the prototypes. The energy function is given as

$$E_{GLVQ} = \frac{1}{2} \sum_{i=1}^{N_V} f(\mu(\boldsymbol{v}_i)) \;\text{, with} \quad \mu(\boldsymbol{v}_i) = \frac{d_{j^+} - d_{j^-}}{d_{j^+} + d_{j^-}} \tag{2.50}$$

where f is a monoton increasing function like the sigmoid or the identical function. The d_{j^+} and d_{j^-} denote the distances between data point $\boldsymbol{v}_i$ and the best matching prototype $\boldsymbol{w}_{j^+} \in W^+ \subset W$ belonging to the same class and the best matching prototype $\boldsymbol{w}_{j^-} \in W^- \subset W$ belonging to any other class, respectively. Note that the numerator of $\mu(\boldsymbol{v}_i)$ is negative, if the classification of $\boldsymbol{v}_i$ is correct. The denominator

acts as a scaling factor ensuring that every term is contained in $(-1, 1)$ to avoid numerical problems (Schneider et al. 2008).

Since the cost function now is differentiable, the update of the prototypes can be derived via stochastic gradient descent learning with respect to the $\boldsymbol{w}_j$:

$$\Delta \boldsymbol{w}_{j^+} = \varepsilon f' \mid_{\mu(\boldsymbol{v}_i)} \cdot \frac{2d_{j^-}}{(d_{j^+} + d_{j^-})^2} \cdot \frac{\partial d_{j^+}}{\partial \boldsymbol{w}_{j^+}} \tag{2.51}$$

$$\Delta \boldsymbol{w}_{j^-} = -\varepsilon f' \mid_{\mu(\boldsymbol{v}_i)} \cdot \frac{2d_{j^+}}{(d_{j^+} + d_{j^-})^2} \cdot \frac{\partial d_{j^-}}{\partial \boldsymbol{w}_{j^-}} \tag{2.52}$$

where ε as usually is a small learning rate. If the squared Euclidean distance is used, i. e. $d(\boldsymbol{v}_i, \boldsymbol{w}_j) = (\boldsymbol{v}_i - \boldsymbol{w}_j)^2$, the derivative of $d(\boldsymbol{v}_i, \boldsymbol{w}_j)$ with respect to $\boldsymbol{w}_j$ yields $\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)/\partial \boldsymbol{w}_j = -2(\boldsymbol{v}_i - \boldsymbol{w}_j)$. The algorithm based on the Euclidean distance is given in algorithm 2.2.3.

Algorithm 2.2.3 – Generalized LVQ (GLVQ)

1. Initialize the labeled codebook vectors $\boldsymbol{w}_j \in W$ randomly
2. Present a labeled input vector $\boldsymbol{v}_i$
3. Determine the closest codebook vector $\boldsymbol{w}_{s^+(\boldsymbol{v}_i)} \in W^+$ according to (2.45)

$$s^+(\boldsymbol{v}_i) = \underset{j' \mid w_{j'} \in W^+}{\operatorname{argmin}} (\boldsymbol{v}_i - \boldsymbol{w}_{j'})^2$$

4. Determine the closest codebook vector $\boldsymbol{w}_{s^-(\boldsymbol{v}_i)} \in W^-$ according to (2.45)

$$s^-(\boldsymbol{v}_i) = \underset{j' \mid w_{j'} \in W^-}{\operatorname{argmin}} (\boldsymbol{v}_i - \boldsymbol{w}_{j'})^2$$

5. Update $\boldsymbol{w}_{s(\boldsymbol{v}_i)^+}$ and $\boldsymbol{w}_{s(\boldsymbol{v}_i)^-}$ according to (2.51) and (2.52)

$$\boldsymbol{w}_{s^+(\boldsymbol{v}_i)}(t+1) = \boldsymbol{w}_{s^+(\boldsymbol{v}_i)^+}(t) - \varepsilon f' \mid_{\mu(\boldsymbol{v}_i)} \cdot \frac{4d_{j^-}}{\left(d_{j^-} + d_{j^+}\right)^2} \cdot (\boldsymbol{v}_i - \boldsymbol{w}_{j^+})$$

$$\boldsymbol{w}_{s^-(\boldsymbol{v}_i)}(t+1) = \boldsymbol{w}_{s^-(\boldsymbol{v}_i)^-}(t) + \varepsilon f' \mid_{\mu(\boldsymbol{v}_i)} \cdot \frac{4d_{j^+}}{\left(d_{j^-} + d_{j^+}\right)^2} \cdot (\boldsymbol{v}_i - \boldsymbol{w}_{j^-})$$

6. Repeat steps 2. – 5. until convergence or manual stop

Note that again – just as for the LVQ2.1 – two prototypes will be updated at each learning step, but now the update is not restricted to the two closest overall prototypes (under certain conditions, see section 2.2.2), yet rather to the closest same-class prototype and the closest not matching prototype. But anyway, for overlapping classes GLVQ optimizes the hypothesis margin (Crammer et al. 2002) and it has been shown, that for this kind of data the behaviour of GLVQ compared to LVQ is more robust, since convergence is ensured, and GLVQ shows better classification results (Sato and Yamada 1998, Hammer and Villmann 2002).

2.2.4 Robust Soft LVQ

The Robust Soft LVQ algorithm (RSLVQ), another advanced learning vector quantizer, was introduced by SEO & OBERMAYER (Seo and Obermayer 2003). Unlike the basic LVQ1 and LVQ2.1 variants, which are based on heuristics, RSLVQ incorporates a statistical model making all assumptions explicit. It is assumed that the probability density $p(\boldsymbol{v})$ of the data points $\boldsymbol{v} \in \mathbb{R}^d$ can be described by a Gaussian mixture model. Every component of the mixture is assumed to generate data which belongs to only one of the N_C classes. The classification itself is based on a winner takes all scheme.

The probability density of all the data points is given by

$$p(\boldsymbol{v}|W) = \sum_{k=1}^{N_C} \sum_{j:y(\boldsymbol{w}_j)=k}^{N_P} p(\boldsymbol{v}|j)P(j) \tag{2.53}$$

where $W = \{(\boldsymbol{w}_j, y(\boldsymbol{w}_j))\}_{j=1}^{N_P}$ is the set of N_P labeled prototype vectors $\boldsymbol{w}_j \in \mathbb{R}^d$ and their assigned class labels $y(\boldsymbol{w}_j)$. $P(j)$ stands for the probability that data points are generated by component j of the mixture and is commonly set to an identical value for all the prototypes. The conditional density $p(\boldsymbol{v}|j)$, which describes the probability that component j is generating a particular data point $\boldsymbol{v}$, is a function of the prototype $\boldsymbol{w}_j$ itself. The density $p(\boldsymbol{v}|j)$ can be chosen to have the normalized exponential form $p(\boldsymbol{v}|j) = K(j) \cdot e^{f(\boldsymbol{v},\boldsymbol{w}_j,\sigma_j^2)}$ where $K(j)$ is the normalization constant and the hyper parameter σ_j^2 the width of component j.

The aim of RSLVQ is to place the prototypes such that a given data set is classified as accurately as possible. Therefore the likelihood ratio

$$L = \prod_{i=1}^{N_V} L(\boldsymbol{v}_i, c(\boldsymbol{v}_i)) \,, \qquad \text{with } L(\boldsymbol{v}_i, c(\boldsymbol{v}_i)) = \frac{p(\boldsymbol{v}_i, c(\boldsymbol{v}_i)|W)}{p(\boldsymbol{v}_i|W)} \tag{2.54}$$

where N_V is the number of data points, has to be maximized. The ratio is built up of the particular probability density $p(\boldsymbol{v}_i, c(\boldsymbol{v}_i)|W)$, that data point $\boldsymbol{v}_i$ is generated

by a mixture component of the correct class $c(\boldsymbol{v}_i)$

$$p(\boldsymbol{v}_i, c(\boldsymbol{v}_i)|W) = \sum_{j:c(\boldsymbol{w}_j)=c(\boldsymbol{v}_i)} p(\boldsymbol{v}_i|j)P(j) \tag{2.55}$$

with the total probability density $p(\boldsymbol{v}_i|W)$

$$p(\boldsymbol{v}_i|W) = \sum_j p(\boldsymbol{v}_i|j)P(j). \tag{2.56}$$

The cost function is given as

$$E_{RSLVQ} = \sum_{i=1}^{N_V} \log\left(\frac{p(\boldsymbol{v}_i, c(\boldsymbol{v}_i)|W)}{p(\boldsymbol{v}_i|W)}\right). \tag{2.57}$$

The learning rules are obtained by a stochastic gradient ascent thereof (Robbins and Monro 1951), so that

$$\Delta\boldsymbol{w}_j = \frac{\varepsilon(t)}{\sigma^2}\begin{cases} (P_{c(\boldsymbol{v})}(j|\boldsymbol{v}) - P(j|\boldsymbol{v}))(\boldsymbol{v}-\boldsymbol{w}_j), & c(\boldsymbol{v}) = y(\boldsymbol{w}_j) \\ -P(j|\boldsymbol{v})(\boldsymbol{v}-\boldsymbol{w}_j), & c(\boldsymbol{v}) \neq y(\boldsymbol{w}_j) \end{cases} \tag{2.58}$$

The variable $\varepsilon > 0$ is again the learning rate which decreases during the training. To ensure convergence certain conditions have to be fulfilled. Assume $\varepsilon(t)$ is the learning rate for update step t, then the constraints $\sum_{t=0}^{\infty}\varepsilon(t) = \infty$ and $\sum_{t=0}^{\infty}\varepsilon^2(t) < \infty$, forcing $\varepsilon(t)$ do decrease slowly but not too slow, have to be kept (Robbins and Monro 1951). The width σ of every component j is assumed to be identical and usually decreases during learning. $P_{c(\boldsymbol{v})}(j|\boldsymbol{v})$ and $P(j|\boldsymbol{v})$ are the assignment probabilities of data sample $\boldsymbol{v}$ to component j within class $y(\boldsymbol{w}_j)$ and independent of the class membership, respectively. Note that the update factors $(P_{c(\boldsymbol{v})}(j|\boldsymbol{v}) - P(j|\boldsymbol{v}))$ and $P(j|\boldsymbol{v})$ act as attracting respectively repulsing forces on the prototypes with correct and incorrect class labels. Contrary to LVQ and GLVQ, all prototypes are updated at one learning step. A window rule is no longer necessary, since the prototypes are not diverging. The width of the active region is regulated by σ. Further details concerning the prototype update (Seo and Obermayer 2003) as well as the adaptation of the hyperparameter σ (Schneider et al. 2008) can be found in the cited references.

2.2.5 Soft Nearest Prototype Classification

SEO, BODE & OBERMAYER proposed a method called Soft Nearest Prototype Classification (SNPC) for the construction of a NPC which is based on a Gaussion mixture

ansatz and which can be interpreted as an annealed version of LVQ (Seo et al. 2003). The algorithm performs a gradient descent on a cost function minimizing the classification error on the training set. For their method SEO ET AL. combined an explicit ansatz for the probability densities of the classes with a criterion for model selection which directly minimizes the rate of misclassification.

Given is a set of N_V training data points $V = \{(\boldsymbol{v}_i, c_i)\}_{i=1}^{N_V}$ and a set of N_P labeled prototypes $W = \{(\boldsymbol{w}_j, y_j)\}_{j=1}^{N_P}$ with $\boldsymbol{v}_i, \boldsymbol{w}_j \in \mathbb{R}^d$. The class labels $c_i = c(\boldsymbol{v}_i)$ and $y_j = y(\boldsymbol{w}_j)$ of the data points $\boldsymbol{v}_i$ respectively prototypes $\boldsymbol{w}_j$ are crisp.

It is assumed that the probability density $p(\boldsymbol{v}_i)$ of the data points $\boldsymbol{v}_i$ can be described by a mixture model: each component j of the mixture generates data points which belong to only one class y_j. The total probability density is given by

$$p(\boldsymbol{v}_i|W) = \sum_{c=1}^{N_C} \sum_{\{j:y_j=c_i\}} p(\boldsymbol{v}_i|j)p(j) \tag{2.59}$$

where N_C is the number of classes, $p(j)$ the probability that data points are generated by a particular component j, and $p(\boldsymbol{v}_i|j)$ the conditional probability that this component j generates a particular data point $\boldsymbol{v}_i$.

Further, there are the restricted probability densities

$$p(\boldsymbol{v}_i, c_i|W) = \sum_{\{j:y_j=c_i\}} p(\boldsymbol{v}_i|j)p(j) \qquad p(\boldsymbol{v}, \bar{c}_i|W) = \sum_{\{j:y_j\neq c_i\}} p(\boldsymbol{v}_i|j)p(j) \tag{2.60}$$

where $p(\boldsymbol{v}_i, c_i|W)$ and $p(\boldsymbol{v}_i, \bar{c}_i|W)$ are the probability densities, that a data point $\boldsymbol{v}_i$ is generated with the correct class label c_i or an incorrect class label $\bar{c}_i$ differing from c_i, respectively.

The aim of SNPC is to minimize the cost function

$$E_{SNPC} = \frac{1}{N_V} \sum_{i=1}^{N_V} \mathrm{lc}((\boldsymbol{v}_i, c_i), W) \tag{2.61}$$

where the local costs

$$\mathrm{lc}((\boldsymbol{v}_i, c_i), W) = \sum_{j=1}^{N_P} \frac{p(\boldsymbol{v}_i, \bar{c}_i|W)}{p(\boldsymbol{v}_i|W)} \tag{2.62}$$

represent the rate of misclassification. During the training these local costs – further abbreviated as lc_i – have to be minimized with respect to the prototypes $\boldsymbol{w}_j$.

If a mixture ansatz with d-dimensional Gaussian components of equal width $\sigma_j^2 = \sigma^2$ and equal strength $p(j) = 1/N_P, \forall j = 1, \ldots, N_P$ is assumed, the conditional probability is obtained by

$$p(\boldsymbol{v}_i|j) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\sigma^2}\right) \tag{2.63}$$

and the local costs can be described by

$$\mathrm{lc}_i = \sum_{j=1}^{N_P} P(j|\boldsymbol{v}_i)(1 - \delta_{c_i,y_j}) \tag{2.64}$$

where

$$P(j|\boldsymbol{v}_i) = \frac{\exp\left(-d(\boldsymbol{v}_i, \boldsymbol{w}_j)/(2\sigma^2)\right)}{\sum_k \exp\left(-d(\boldsymbol{v}_i, \boldsymbol{w}_k)/(2\sigma^2)\right)} \tag{2.65}$$

is the posterior probability, that the data point $\boldsymbol{v}_i$ was generated by component j, and the Kronecker symbol δ_{c_i,y_j} is one if $c_i = y_j$ and zero otherwise. The update rule for the prototypes derived by stochastic gradient descent on the local costs is then given as

$$\Delta\boldsymbol{w}_j = -\frac{\varepsilon}{2\sigma^2} P(j|\boldsymbol{v}_i)(1 - \delta_{c_i,y_j} - \mathrm{lc}_i)\frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j}. \tag{2.66}$$

with the learning rate ε.

Since only data points within a particular area of the input space contribute to the prototype update, a window rule can be applied to accelerate the learning process. For SNPC this active area is specified by $0 \leq \mathrm{lc_i}(1 - \mathrm{lc_i}) \leq 0.25$.

Once the prototypes are determined, new data points $\boldsymbol{v}_k$ can be classified according to

$$y = \mathrm{argmax}_{y'} \sum_{j:y_j=y'} P(j|\boldsymbol{v}_k). \tag{2.67}$$

2.2.6 Fuzzy Soft Nearest Prototype Classification

NPCs realize a crisp classification because of the prototypes' unique class dependence. This has the disadvantage, that overlapping data cannot be described appropriately. In (Villmann, Schleif and Hammer 2006) VILLMANN ET AL. established a new learning scheme called Fuzzy Soft Nearest Prototype Classification (FSNPC), which utilizes fuzzy prototype vectors. The FSNPC is an extension of the SNPC and is also based on the Gaussian mixture model. Formerly crisp class assigments y_j have been replaced by fuzzy prototype labels $\boldsymbol{y}_j$, which indicate the proportionate responsibilities of the weight vectors to all classes N_C with $\sum_{l=1}^{N_C} y_j^l = 1$ and $y_j^l \geq 0$. During the training these labels have to be adjusted to represents the prospective class assignments with respect to the new prototype positions.

Since the crisp class information for the prototypes assumed in the learning dynamic of SNPC now no longer is available, a corresponding learning scheme has

been derived as

$$\Delta \boldsymbol{w}_j = -\frac{\varepsilon}{2\sigma^2} P(j|\boldsymbol{v}_i)(1 - y_j^{c_i} - \mathrm{lc}_i) \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j} \tag{2.68}$$

where the local cost function has been changed to

$$\mathrm{lc}(\boldsymbol{v}_i, c_i) = \sum_{j=1}^{N_P} P(j|\boldsymbol{v}_i)(1 - y_j^{c_k}). \tag{2.69}$$

For the special case of $\boldsymbol{y}_j$ resembling a crisp assignment to one class, equation (2.69) is equivalent to the local cost function of the SNPC (2.64).

Parallely to the adaption of the prototypes, their fuzzy labels can be optimized according to

$$\Delta \boldsymbol{y}_j = -\varepsilon_y P(j|\boldsymbol{v}_k) \tag{2.70}$$

followed by a subsequent normalization of the fuzzy class labels $\boldsymbol{y}_j$.

As for the SNPC there is a window rule specifying the active region for the prototype update (Villmann, Schleif and Hammer 2006): By denoting $T = P(j|\boldsymbol{v}_i)(1 - y_j^{c_i} - \mathrm{lc}_i)$ in equation (2.68) and rewriting it to $T_0 = (T_{lc} - T_{y_j^{c_i}}) \cdot \Pi(y_j^{c_i})$ with $T_{lc} = \mathrm{lc}_i(1 - \mathrm{lc}_i)$ and $T_{y_j^{c_i}} = y_j^{c_i}(1 + y_j^{c_i})$, it can be shown that $-2 \leq T_0 \leq 0.25$ since $0 \leq T_{lc} \leq 0.25$ and $T_{y_j^{c_i}} \leq 1$. The term $\Pi(y_j^{c_i})$ is given by

$$\Pi(y_j^{c_i}) = \frac{\exp(-d(\boldsymbol{v}_i, \boldsymbol{w}_j)/(2\sigma^2)}{\sum_{j'}(1 - y_j^{c_i} - y_{j'}^{c_i})/\exp(-d(\boldsymbol{v}_i, \boldsymbol{w}_{j'})/(2\sigma^2)}. \tag{2.71}$$

The absolute value of T_0 has to be significantly different from zero to have a valuable contribution in the update rule. This yields the window condition $0 \ll |T_0|$, which can be obtained by balancing the local loss lc_i and the value of the assignment variable $\boldsymbol{y}_j$.

As intended for NPCs, unknown data samples are matched to the closest prototype. But since now each prototypes represents different classes proportionately, the data samples are also assigned to different classes.

2.3 Distance measures

Clustering and classification of objects can be seen as partitioning of objects or data into smaller subsets in a way that those objects are grouped together which are similar to each other. Depending on the characteristics of the data it is important to

chose the most appropriate similarity measure. All cluster and classification algorithms described in this work utilize prototypes. Therefore all of these algorithms require the calculation of the (dis)similarities or distances between prototypes and data points. In this section a short summary of different distance measures is given.

For metric data most commonly the Euclidean metric is applied, while other metrics like for example the Mahalanobis or Minkowski distance, the Sobolev norm or inner product might also be suited.

Functional data on the other hand requires a dissimilarity measure which takes the characteritics of the usually very high-dimensional data vectors $\boldsymbol{x}$ into account. The vector components of functional data are spatially correlated, whereas for common Euclidean vectors the vector dimensions are treated independently. If these functions are assumed to be positive with finite $\mathfrak{L}_1$-norm, the dissimilarity between such functions can be evaluated by (generalized) divergence measures taking into account the functional character (Villmann and Haase 2011). In Section 2.3.2 the basic properties along with a selection of different divergences are presented.

And finally, for non-metric data, e. g. text documents or music, the similarity can be approximated by the normalized information distance, which is based on the Kolmogorov complexity.

2.3.1 Euclidean metric and variants

The concept of metric spaces was introduced by FRÉCHET (Fréchet 1906), who stated a set of axioms to define a metric. If it is assumed, that $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in M$, where M is a non-empty set, then the function d is a metric on M if the following holds:

1. $d(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ non-negativity
2. $d(\boldsymbol{x}, \boldsymbol{x}) = 0$ identity/reflexivity
3. $(d(\boldsymbol{x}, \boldsymbol{y}) = 0) \Rightarrow (\boldsymbol{y} = \boldsymbol{x})$ definiteness
4. $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$ symmetry
5. $d(\boldsymbol{x}, \boldsymbol{y}) + d(\boldsymbol{y}, \boldsymbol{z}) \geq d(\boldsymbol{x}, \boldsymbol{z})$ triangle inequality

The function $d(\boldsymbol{x}, \boldsymbol{y})$ is often called *distance function* or simply *distance*.

As mentioned before, for clustering or classification usually the (squared) Euclidean distance is used, which itself is a special form of the Minkowski distance

$$d_{Mink}(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{d} \mid x_i - y_i \mid^p \right)^{\frac{1}{p}}, \quad \text{with } p \geq 1 \tag{2.72}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are d-dimensional vectors in $\mathbb{R}^d$. For the special case of $p = 2$ the Euclidean distance is obtained. Further special variants are the Manhatten distance for $p = 1$ and the Maximum norm for $p \to \infty$:

$$d_{Euclid}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{d} \mid x_i - y_i \mid^2} \tag{2.73}$$

$$d_{Manh}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{d} |x_i - y_i| \tag{2.74}$$

$$d_{Max}(\boldsymbol{x}, \boldsymbol{y}) = \max_{i=1,..,d} |x_i - y_i| \,. \tag{2.75}$$

Another useful class of measures for comparing data sets with each other are metrics defined by the bilinearform (quadratic form)

$$d(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T \Lambda (\boldsymbol{x} - \boldsymbol{y}) \tag{2.76}$$

with Λ being a positive definite matrix. A famous example is the Mahalanobis distance. This measure, introduced by MAHALANOBIS (Mahalanobis 1930), is based on the covariance C of the data and is scale invariant. It is given as

$$d_{Maha}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^T C^{-1} (\boldsymbol{x} - \boldsymbol{y})} \tag{2.77}$$

where $\Lambda = C^{-1}$. If the covariance matrix C is the identity matrix, the Mahalanobis distance d_{Maha} is equivalent to the Euclidean distance (2.73), and if C is diagonal with positive entries the scaled Euclidean distance is obtained. This last distance can also be formulated as

$$\begin{aligned} d^{\lambda}_{Euclid}(\boldsymbol{x}, \boldsymbol{y}) &= \sqrt{(\boldsymbol{\lambda} \circ (\boldsymbol{x} - \boldsymbol{y}))^2} \\ &= \sqrt{(\boldsymbol{\lambda} \circ \boldsymbol{x} - \boldsymbol{\lambda} \circ \boldsymbol{y})^2} \end{aligned} \tag{2.78}$$

where the d-dimensional parameter $\boldsymbol{\lambda}$ with the constraints $\lambda_i > 0$ and $\sum_{i=1}^{d} \lambda_1 = 1$ is an integral component of the metric and $\cdot \circ \cdot$ denotes the Hadamard product, which implies element-wise multiplication of the vector components.

2.3.2 Divergences

Divergences measure the similarity between two densities, where for the densities $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ with $\boldsymbol{x} \in V$ the constraints $0 \leq p(\boldsymbol{x}) \leq 1$ and $0 \leq q(\boldsymbol{x}) \leq 1$ are valid and the weigths $W(p) = \int p(\boldsymbol{x})\, d\boldsymbol{x}$ and $W(q) = \int p(\boldsymbol{x})\, d\boldsymbol{x}$ equal 1. If these conditions are relaxed in the sense that $W(p)$ and $W(q)$ are only required to be positive, than divergences can also be used to determine the similarity between positive

measures, i. e. non-negative integrable measure functions. Typical functional data are histograms or high-dimensional spectral data with spatially correlated vector components.

Compared to the metric axioms the following properties are valid:

1. $D(p \parallel q) \geq 0$ non-negativity
2. $D(p \parallel p) = 0$ identity/reflexivity
3. $D(p \parallel q) = 0 \Rightarrow p \equiv q$ definiteness
4. *symmetry not required*
5. *triangle inequality has not to be fulfilled*
6. convex with respect to the first argument

Therefore, although divergences are not a metric, since only the first three metric axioms are fullfilled, they can still be used as a similarity measure.

According to the classification given in CICHOCKI ET AL. (Cichocki and Amari 2010, Cichocki et al. 2009) it can be distinguished between at least three main classes of divergences emphasizing different properties:

(i) Bregman-divergences

(ii) Csiszár's f-divergences

(iii) γ-divergences

Exemplary from the huge field of divergences only a selection of well-known measures will be presented. Since the data considered in latter examples are discrete vector representations of functionals, the following divergences are given in their discrete form where integrals are replaced by sums over the vector components. To be conform with the notation used throughout this work, the functionals $p(x)$ and $q(x)$ are now denoted as $\boldsymbol{v}$ and $\boldsymbol{w}$ respectively, corresponding to data sample and prototype, which are now functions.

η-Divergence (i)

$$D_\gamma(\boldsymbol{v} \parallel \boldsymbol{w}) = \sum_{k=1}^{d} \left(v_k^\eta + (\eta - 1) w_k^\eta - \eta v_k w_k^{\eta - 1} \right) \tag{2.79}$$

The η-divergence ist a member of the class of Bregman divergences and for $\eta = 2$ it

is equivalent with the Euclidean distance. (Nielson and Nock 2009)

β-Divergence (i)

$$D_\beta(\boldsymbol{v} \parallel \boldsymbol{w}) = \sum_{k=1}^{d} \left(v_k \cdot \frac{v_k^{\beta-1} - w_k^{\beta-1}}{\beta - 1} \right) - \sum_{k=1}^{d} \frac{v_k^{\beta} - w_k^{\beta}}{\beta} \tag{2.80}$$

Another representative of the Bregman divergences is the β-divergence which for $\beta \to 1$ approximates the Kullback–Leibler divergence. (Eguchi and Kano 2001, Cichocki et al. 2009)

Generalized Rényi Divergence (ii)

$$D_\alpha^{GR}(\boldsymbol{v} \parallel \boldsymbol{w}) = \frac{1}{\alpha - 1} \log \left[1 + \sum_{k=1}^{d} \left(v_k^{\alpha} w_k^{1-\alpha} - \alpha v_k + (\alpha - 1) w_k \right) \right] \tag{2.81}$$

The Rényi Divergence from Csiszár's f-divergences approximates for the special case of $\alpha \to 1$ the Kullback–Leibler divergence. (Rényi 1961, Rényi 1970)

γ-Divergence (iii)

$$D_\gamma(\boldsymbol{v} \parallel \boldsymbol{w}) = log \left[\frac{\left(\sum_{k=1}^{d} v_k^{\gamma+1} \right)^{\frac{1}{\gamma(\gamma+1)}} \cdot \left(\sum_{k=1}^{d} w_k^{\gamma+1} \right)^{\frac{1}{\gamma+1}}}{\sum_{k=1}^{d} \left(v_k \cdot w_k^{\gamma} \right)^{\frac{1}{\gamma}}} \right] \tag{2.82}$$

The γ-Divergence as a representative of the class of γ-divergences also approximates the Kullback–Leibler divergence for $\gamma \to 0$. (Fujisawa and Eguchi 2008)

Cauchy–Schwarz Divergence (iii)

$$D_{CS}(\boldsymbol{v} \parallel \boldsymbol{w}) = \frac{1}{2} log \left(\frac{\left(\sum_{k=1}^{d} v_k^2 \right)^{\frac{1}{2}} \cdot \left(\sum_{k=1}^{d} w_k^2 \right)^{\frac{1}{2}}}{\sum_{k=1}^{d} \left(v_k \cdot w_k \right)} \right) \tag{2.83}$$

The Cauchy–Schwarz divergence, which can be approximated by the γ-Divergence with $\gamma \to 1$, also belongs to class of γ-divergences. (Principe et al. 2000)

Generalized Kullback–Leibler Divergence (i, ii, iii)

$$D_{GKL}(\boldsymbol{v} \parallel \boldsymbol{w}) = \sum_{k=1}^{d} \left(v_k \log \left(\frac{v_k}{w_k} \right) - (v_k - w_k) \right) \tag{2.84}$$

The Kullback–Leibler divergence is exceptional in the sense, that it can be approximated by representatives of each class. (Kullback and Leibler 1951, Cichocki et al. 2009)

In (Villmann and Haase 2011) VILLMANN & HAASE present a survey of many further divergences together with a complete listing of the respective Fréchet derivatives, the derivatives for relevance learning and hyper-parameter learning, which will be of interest in chapter 3.

2.3.3 Kernel distances

Kernel distances became popular by the introduction of Support Vector Machines (SVM) as powerful classifier systems (Schoelkopf and Smola 2002). The basic idea is to map the data from a data space V into a possibly infinite-dimensional functional feature space H, whose mathematical structure is a Hilbert space. The data mapping is realized by a mapping function Φ, which uniquely corresponds to a so-called kernel $\kappa_\Phi(\boldsymbol{x}, \boldsymbol{y})$. This kernel is a positive definite function on $V \times V$ (Aronszajn 1950) and determines the inner product in the Hilbert space H, hence, it defines a metric d_H. This metric is called kernel metric and measures the dissimilarity between the mapped data. If the kernel is continuous and universal, then the mapping is injective, i. e. it generates a unique representation of the data in the Hilbert space (Steinwart 2001). Usually the topological structure of the Hilbert space has a wider richness of shape then the original data space and therefore allows a better adapted classifier system for SVMs.

The main advantage of kernel approaches is that the dissimilarity in the Hilbert space can be calculated without explicit knowledge of the mapping function Φ. Only the kernel

$$d_H(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\kappa_\Phi(\boldsymbol{x}, \boldsymbol{x}) - 2\kappa_\Phi(\boldsymbol{x}, \boldsymbol{y}) + \kappa_\Phi(\boldsymbol{y}, \boldsymbol{y})} \tag{2.85}$$

is necessary for the calculations and can directly be implemented into batch learning schemes like Kernel SOM (Boulet et al. 2008, Hulle 2009) and Kernel FCM (Yang et al. 2011). Other approaches use approximation techniques to incorporate kernel distances in online learning for classification (Schleif et al. 2011, Qin and Suganthan 2004a, Qin and Suganthan 2004b). A very recent approach is utilization of differentiable kernels, such that online learning can be ensured without any kernel approximation (Villmann and Haase 2012).

2.3.4 Normalized Information Distance

The Normalized Information Distance (NID) is a universal distance measure for objects of all kinds like music (Cilibrasi and Vitányi 2005, Li and Sleep 2004), gene sequences (Fisher et al. 2010, Cilibrasi and Vitányi 2005), text (Geweniger et al. 2009) and so on and is calulated according to

$$NID_{x,y} = \frac{Z_{x,y} - min(Z_x, Z_y)}{max(Z_x, Z_y)}. \tag{2.86}$$

This distance is a metric (Bennett et al. 1998), in particular it is symmetric, and based on the Kolmogorov complexity, which is the minimal description length (MDL) of the objects (Bennett et al. 1998, Vitányi and Li 2000). To calculate the normalized information distance (NID), e. g. for text data, the minimal description length Z_x of a single text document x and the respective $Z_{x,y}$ for pairwise combined documents x and y have to be considered. For objects with string representation, the MDL is estimated by the compression length z according to a given standard compression scheme or algorithm like for example the Lempel-Ziv-Markow algorithm (LZMA) and objects x and y are combined by concatenation. The NID is then approximated by the normalized compression distance (NCD) (Cilibrasi and Vitányi 2005):

$$NCD_{x,y} = \frac{z_{xy} - min(z_x, z_y)}{max(z_x, z_y)}. \tag{2.87}$$

Due to technical reasons NCD violates the symmetry property in the sense that $NCD_{x,y} - NCD_{y,x} = \delta$ with $0 < \delta \ll 1$ and usually it is a magnitude smaller than the NID values.

2.4 Validation measures for fuzzy clustering and classification

The validation of a clustering or classification solution is an ill-posed problem which has to be considered carefully. Some question to answer in order to select the appropriate measure might involve:

- What has to be evaluated: a clustering solution, a classification solution or the performance of an algorithm?
- Does a reference solution exist? In that case the clustering or classification could be compared to the known solution.
- What kind of data are we dealing with? Metric or non-metric? ...

To evaluate a prototype based clustering the position of the prototypes with respect to the position of the represented data points could be examined. There are measures aiming to assess values thereof in terms of separation and compactness. Well known examples like the *Partition Entropy* by BEZDEK (Bezdek 1974b, Bezdek 1974a) and the *Xie Beni Index* (Xie and Beni 1991, Pal and Bezdek 1995) are presented in the following section.

A way to evaluate a classification is to classify unlearned data points and compare their assigments to the known class labels and calculate the relative rate of mismatches.

If a known reference solution, either obtained by another clustering or since the data is artificially designed, is available, then the *Rand Index* (Rand 1971) or similar measures as described in section 2.4.2 can be employed.

Two measures — *Cohen's Kappa* (Cohen 1960) and *Fleiss' Kappa* (Fleiss 1981) — for the comparison of two or more classifiers or classifications are described in section 2.4.3. Both measures take the agreement by chance into account, yet for the evaluation of clusterings they are suitable only to an limited extent, since all possible cluster permutations have to be considered.

All of the subsequently presented measures are also available for fuzzy data. Further details and restrictions can be found in the respective sections.

2.4.1 Measures based on separation and compactness

These measures are usually used to assess the optimal number of prototypes. Thereby a clustering is assumed to be *good*, if the clusters are well separated and very compact (Žalik and Žalik 2010, Kim et al. 2004). For tests data sets or those well known from former examinations where the number of clusters is known, these measures might as well be used for he evaluation of an obtained clustering. Following an incomplete list of a selection of measures based on separation and compactness. Since all of the presented measures take the assignments $u_j \boldsymbol{v}_i$, abbreviated as u_{ij}, which indicate the magnitude to which each data point $\boldsymbol{v}_i$ is assigned to prototype $\boldsymbol{w}_j$, into account, they are suitable for fuzzy clusterings.

Partition Coefficient

$$PC = \frac{1}{N_V} \sum_{j=1}^{N_P} \sum_{i=1}^{N_V} u_{ij}^m \tag{2.88}$$

Proposed by BEZDEK measures the Partition Coefficient the compactness of a clustering and is based on the fuzzy memberships only. The crisper the values of u_{ij},

the higher is the compactness and the index value increases. (Bezdek 1974b)

Partition Entropie

$$PE = -\frac{1}{N_V}\sum_{j=1}^{N_P}\sum_{i=1}^{N_V} u_{ij}\cdot\log_m(u_{ij}) \tag{2.89}$$

The Partition Entropy was also introduced by BEZDEK and is also a measure of compactness only. Yet contrary to the Partition Coefficient now a low index value is desired. (Bezdek 1974b, Bezdek 1974a)

Fukuyama & Sugeno

$$FS = \sum_{j=1}^{N_P}\sum_{i=1}^{N_V} u_{ij}^m d(\boldsymbol{v}_i,\boldsymbol{w}_j) - \sum_{j=1}^{N_P}\sum_{i=1}^{N_V} u_{ij}^m d(\boldsymbol{w}_j,\bar{\boldsymbol{w}})\,, \qquad \bar{\boldsymbol{w}} = \sum_{j=1}^{N_P}\boldsymbol{w}_j/c \tag{2.90}$$

The validity function proposed by FUKUYAMA & SUGENO combines the compactness and the separation of a cluster solution and is desired to obtain a minimum. (Fukuyama and Sugeno 1989)

Xie & Beni

$$XB = \frac{\sum_{j=1}^{N_P}\sum_{i=1}^{N_V} u_{ij}^m d(\boldsymbol{v}_i,\boldsymbol{w}_j)}{\mathtt{min}_{j\neq l} d(\boldsymbol{w}_j,\boldsymbol{w}_l)} \tag{2.91}$$

This index stats that a good clustering is obtained by minimizing the compactness and maximizing the separation. Therefore the aim is to achieve a low value for the index. (Xie and Beni 1991, Pal and Bezdek 1995)

Problems with these indexes arise for overlapping discrete data, since the separation value will be poor. Also, if dealing with cluster assignments, where each cluster is represented by more than one prototype, these measures can not be used. An alternative, the CONN-Index was introduced by TAŞDEMIR & MERÉNYI (Taşdemir and Merényi 2011). But since as of now this measure is not yet available for fuzzy data it is not presented in this section. Yet it is subject to further work and first results are already achieved.

2.4.2 Rand index and related measures

A further criterion for the evaluation of cluster or classification solutions[3] is the Rand Index as proposed by RAND (Rand 1971). This measure compares objects

[3]Within this section the term *cluster* will be used synonymously for *cluster* and *class*.

pairs, counts their agreements in terms of class memberships and calculates the index value. Thereby the number of clusters or classes for both solutions does not have to be the same. There are four different possibilities to define pairwise class memberships:

a - an object pair belongs to one cluster in the first solution and also belongs to one cluster in the second solution

b - an object pair belongs to one cluster in the first solution, but to different clusters in the second solution

c - an object pair belongs to different clusters in the first solution, but to only one cluster in the second solution

d - an object pair belongs to different clusters in the first solution, and also to different clusters in the second solution

Based on the number of occurances of a to d the following indexes can be calculated:

Rand Index

$$RI = \frac{a+d}{a+b+c+d} \tag{2.92}$$

The Rand index sets the number of pairwise agreements (both in one cluster or both in different clusters) in relation to the number of possible pairs. The resulting index is value in $[0,1]$, where 1 implies complete agreement.

In (Campello 2007) CAMPELLO derived a Fuzzy Rand Index using *t-norms* and *t-conorms* to obtain valid values for a to d based on fuzzy assignments. Detailed descriptions can be found in (Campello 2007). A major drawback of the Fuzzy Rand Index is that it can only be used to compare a discrete with an fuzzy solution. It is not suitable for comparing two fuzzy solutions with each other, since in the case of perfect agreement, the Fuzzy Rand Index does not result in 1, which is caused by the use of the *t-norms*. Further is has to be paid attention to the choice of the *t-norm*. As considered in (Zühlke et al. 2009), different different *t-norms* like Minimum, Sum or Łukasiewicz leads to different values for a to d and therefore to varying results for the Fuzzy Rand Index.

Adjusted Rand Index (Hubert and Arabie 1985)

$$ARI = \frac{a - \frac{(a+c)(a+b)}{d}}{\frac{(a+c)+(a+b)}{2} - \frac{(a+c)(a+b)}{d}} \tag{2.93}$$

Jaccard Coefficient (Jain and Dubes 1988)

$$J = \frac{a}{a+b+c} \tag{2.94}$$

Fowlkes-Mallows Index (Fowlkes and Mallows 1983)

$$FM = \frac{a}{\sqrt{(a+b)(a+c)}} \tag{2.95}$$

Minkowski Measure (Sokal 1977)

$$M = \sqrt{\frac{b+c}{b+a}} \tag{2.96}$$

Γ **Statistics** (Jain and Dubes 1988)

$$\Gamma = \frac{Ma-(a+b)(a+c)}{\sqrt{(a+b)(a+c)(M-(a+b))(M-(a+c))}} \quad , M = N_V(N_V-1)/2 \tag{2.97}$$

Each of these indexes shows some special characteristics. Further details can be found in the before mentioned article by CAMPELLO (Campello 2007) and also in the literature listed there.

2.4.3 Cohen' Kappa and Fleiss' Kappa

The Kappa Statistic as introduced by COHEN in 1960 (Cohen 1960) was originally intended to assess the reliability of a classification system. The Kappa Statistic or also Kappa Coefficient is a correlation-like coefficient of pairwise observed agreement and the expected agreement by chance. For classification tasks the Kappa Coefficient indicates the ratio of the error generated by a classification algorithm to the error of a complete random classification.

A major drawback of Cohen's Kappa Coefficient is, that this measure is only capable of comparing two classifiers with each other. An advanced method was

proposed by FLEISS (Fleiss 1981). The calculation of Fleiss' Kappa is a direct extension of Cohen's Kappa, yet takes more than two classifiers into account. If these measures are used to compare different cluster solutions with each other, it has to be assured, that all solutions have the same number of clusters and that all possible permutations are considered.

The following section briefly describes the two inter-rater agreement measures. Thereafter the fuzzy variants are presented, whereas Fuzzy Cohen's Kappa was proposed by DOU ET AL. (Dou et al. 2007) and Fuzzy Fleiss' Kappa is a joint work of ZÜHLKE, GEWENIGER, HEIMANN & VILLMANN based on the discrete Fleiss' Kappa and Fuzzy Cohen's Kappa (Zühlke et al. 2009).

Original Cohen's and Fleiss' Kappa

Cohen's Kappa κ_C is a statistical measure of the inter-rater agreement of only *two* discrete classifiers C_1 and C_2. It is given by

$$\kappa_C = \frac{P_o - P_e}{1 - P_e} \tag{2.98}$$

where P_o is the observed relative agreement between the two classifiers and can be counted given a contingency table of the respective outcomes:

$$P_o = \frac{1}{N_V} \sum_{i=1}^{N_V} \sum_{k=1}^{N_C} u_k^{C_1}(v_i) \cdot u_k^{C_2}(v_i). \tag{2.99}$$

P_e is the expected agreement by chance, i. e. it is the expected value of the joined event of classifier C_1 and C_2 classifying a data point v_i to the same class. Under the assumption of independent classifiers C_1 and C_2, P_e can be calculated as follows:

$$P_e = \sum_{k=1}^{N_C} \sum_{u_k^{C_1}=0}^{1} \sum_{u_k^{C_2}=0}^{1} p_k^{C_1} \cdot p_k^{C_2} \left(u_k^{C_1} \cdot u_k^{C_2} \right). \tag{2.100}$$

The discrete assignment function $u_k^{C_m}(v_i) \in \{0, 1\}$ with $m \in \{1, 2\}$ assigns each data point v_i to exactly one of the N_C classes and the values $p_k^{C_1}$ and $p_k^{C_2}$ are the margin probabilities (densities) $p_k^{C_m} = \frac{1}{N_V} \sum_{i=1}^{N_V} u_k^{C_m}(v_i)$.

As mentioned above the calculation of Fleiss' Kappa κ_F is a direct expansion of Cohen's Kappa for $M > 2$ classifiers. For the discrete classification P_o^M is the

counted relative agreement between M classifiers:

$$P_o^M = \frac{1}{N_V} \sum_{i=1}^{N_V} \sum_{k=1}^{N_C} \prod_{m=1}^{M} u_k^{C_m}(v_i). \tag{2.101}$$

The respective value for P_e calculates as follows

$$P_e^M = \sum_{k=1}^{N_C} \sum_{u_k^{C_1}=0}^{1} \cdots \sum_{u_k^{C_M}=0}^{1} \prod_{m=1}^{M} p_k^{C_m} \cdot u_k^{C_m}. \tag{2.102}$$

The value of Fleiss' Kappa itself is calculated analogously to eq. (2.98)

$$\kappa_F = \frac{P_o^M - P_e^M}{1 - P_e^M} \tag{2.103}$$

For both Kappa the relation $\kappa \leq 1$ is valid and the values are interpreted according to the scheme given in Tab. 2.1 (Sachs 1992).

κ **- value**	**meaning**
$\kappa < 0$	poor agreement
$0 \leq \kappa \leq 0.2$	slight agreement
$0.2 < \kappa \leq 0.4$	fair agreement
$0.4 < \kappa \leq 0.6$	moderate agreement
$0.6 < \kappa \leq 0.8$	substantial agreement
$0.8 < \kappa \leq 1.0$	perfect agreement

Table 2.1: ***Interpretation of*** κ ***- values*** *- Values below zero indicate poor or accidential agreement, whereas values above zero indicate an agreement of some degree. (Sachs 1992)*

Fuzzy variants of Cohen's and Fleiss' Kappa

In case of fuzzy classifiers the discrete values u_k are turned into continuous values μ_k with $\sum_{k=1}^{N_C} \mu_k(v) = 1$ and $\mu_k(v) \geq 0$. The fuzzy variant of κ_C is derived according to DOU ET AL. (Dou et al. 2007), where the products $u_k^{C_1} \cdot u_k^{C_2}$ in eqs. (2.99) and (2.100) have been replaced by a *logical* AND-operator without loss. Transfering this idea to fuzzy classifiers yields for P_o

$$P_o = \frac{1}{N_V} \sum_{i=1}^{N_V} \sum_{k=1}^{N_C} \left(\mu_k^{C_1}(v_i) \bigwedge \mu_k^{C_2}(v_i) \right) \tag{2.104}$$

Since the discrete values u_k have been replaced by continuous values μ_k, the sums over the discrete values for the u_k in eq. (2.100) become integrals over μ_i. Thus P_e can be calculated analogously

$$P_e = \sum_{k=1}^{C} \int_0^1 \int_0^1 p_k^{C_1} \cdot p_k^{C_2} \cdot \left(\mu_k^{C_1} \bigwedge \mu_k^{C_2} \right) d\mu_k^{C_2} d\mu_k^{C_1} \tag{2.105}$$

where the values $p_k^{C_1} = p(\mu_k^{C_1})$ and $p_k^{C_2} = p(\mu_k^{C_2})$ are the probability distributions of $\mu_k^{C_1}(v_i)$ and $\mu_k^{C_2}(v_i)$, respectively. The obtained values for P_o and P_e can be inserted into equation (2.98) to calculate the Fuzzy Cohen's Kappa value $\widehat{\kappa}_C$.

Yet, while for discrete classifiers the product can equivalently be replaced by the binary AND-operator, in the fuzzy case this operator cannot be uniquely determined. There exist a number of possibilities based on the concept of *t-norms* (Hammer and Villmann 2007). In general a function $\top : [0,1]^2 \rightarrow [0,1]$ is called a *t-norm*, if the following holds:

- $\top(a,1) = a$ (neutral element)
- $a \leq b \Rightarrow \top(a,c) \leq \top(b,c)$ (monotonicity)
- $\top(a,b) = \top(b,a)$ (commutativity)
- $\top(a,\top(b,c)) = \top(\top(a,b),c)$ (associativity)

Obviously, the definition does not determine a unique norm. Examples are

- the minimum norm $\top_{min}(a,b) = min\{a,b\}$
- the product norm $\top_{prod}(a,b) = a \cdot b$
- the Łukasiewicz norm $\top_{Luka}(a,b) = max\{0, a+b-1\}$

In (Zühlke et al. 2009) we have shown that the most appropriate and reliable norm is the minimum norm. With this choice the most reliable results can be achieved and the interpretation of the κ-value as given in table (2.1) can still be considered.

Rewriting equations eqs. (2.104) and (2.105) using the *t-norm* notation yields

$$P_o = \frac{1}{N_V} \sum_{i=1}^{N_V} \sum_{k=1}^{N_C} \top(\mu_k^{C_1}(v_i), \mu_k^{C_2}(v_i)) \tag{2.106}$$

$$P_e = \sum_{k=1}^{N_C} \int_{\mu_k^{C_1}=0}^{1} \int_{\mu_k^{C_2}=0}^{1} p(\mu_k^{C_1}) \cdot p(\mu_k^{C_2}) \cdot \left(\top(\mu_k^{C_1}, \mu_k^{C_2}) \right) d\mu_k^{C_2} d\mu_k^{C_1} \tag{2.107}$$

Also in (Zühlke et al. 2009) we derived the fuzzy variant of Fleiss' kappa. Based on eqs. (2.101) and (2.102) and again using the concept of *t-norms* to compare the fuzzy agreements, the values of P_o^M and P_e^M can be calculated as

$$P_o^M = \frac{1}{N_V} \sum_{i=1}^{N_V} \sum_{k=1}^{N_C} \top(\mu_k^{C_1}(v_i), \ldots, \mu_k^{C_M}(v_i)) \tag{2.108}$$

$$P_e^M = \sum_{k=1}^{N_C} \int_{\mu_k^{C_1}=0}^{1} \ldots \int_{\mu_k^{C_M}=0}^{1} \left(\prod_{m=1}^{M} p(\mu_k^{C_m}) \right) \cdot \left(\top(\mu_k^{C_1}, ..., \mu_k^{C_M}) \right) d\mu_k^{C_M} ... d\mu_k^{C_1} \tag{2.109}$$

As before the value of the fuzzy version of Fleiss' Kappa Coefficient $\widehat{\kappa}_F$ is computed by inserting P_o^M and P_e^M into eq. (2.103).

Chapter based on:

Tina Geweniger, Marika Kästner, and Thomas Villmann – *"Optimization of parametrized divergences in fuzzy c-means,"* in M. Verleysen (ed.), Proc. of European Symposium on Artificial Neural Networks (ESANN 2011), d-side publications, Evere, Belgium, 11-16, 2011.

Tina Geweniger, Dietlind Zühlke, Barbara Hammer, and Thomas Villmann – *"Median fuzzy c-means for clustering dissimilarity data,"* Neurocomputing, vol. 73, no. 7-9, pp. 1109-1116, 2010.

Tina Geweniger, Dietlind Zühlke, Barbara Hammer, and Thomas Villmann – *"Fuzzy Variant of Affinity Propagation in Comparison to Median Fuzzy c-Means,"* in J. Principe (ed.), Advances in Self-Organizing Maps - Proc. of the Workshop on Self-Oranizing Maps (WSOM), LNCS 5629, Springer, pp. 72-79, 2009.

Chapter 3

Fuzzy clustering for non-standard metrics

Abstract

In this chapter some extensions and further enhancements to known prototype based clustering algorithms are presented. Of special interest are non-metric distance measures for fuzzy clusterings. Frequently, the Euclidean distance is used to measure the distance between data points and prototypes. Yet, if the data is not embedded in a metric space, algorithms based on similarities only have to be employed. In the following a median based variant of the well known Fuzzy c-Means and a fuzzy interpretation of Affinity Propagation are proposed. Further, for the special case of functional data, the usability of divergences as distance measure for the Fuzzy c-Means has been investigated. Additionaly, a relevance parameter to weight the input dimensions was introduced and an appropriate update rule derived.

Frequently, clustering of objects is seen as partitioning of objects or data into smaller subsets such that those objects are grouped together, which have similar semantical meanings, characteristics, or behavior. The clustering approaches may differ in various aspects like flat or hierarchical structure, crisp or soft (fuzzy) cluster assignments, visualization driven approaches, etc. Yet, the similarity is frequently difficult to capture. For metric data usually the Euclidean metric is applied. However, this favored choice may be inappropriate as it has been shown for functional data analysis by RAMSAY & SILVERMANN (Ramsay and Silverman 2006). Or refer to INOKUCHI & MIYAMOTO, who introduced divergences to be used with the Fuzzy c-Means algorithm to obtain fuzzy clusters of discrete data (Inokuchi and

Miyamoto 2008). Generally, clustering is an ill–posed problem and chosing a adequate similarity measure is crucial and heavily influencing the clustering result as well as the number of meaningful clusters (Gordon 1998, Sen and Dave 2000). A further problem arising in clustering is the choice of an appropriate quality criteria for a given cluster solution. There exists a large number of quality measures for cluster verification and validation (Pal and Bezdek 1995). Prominent examples are the Fishers information relating the intra- and the inter-cluster correlation, the κ-statistics based on χ^2-statistics or Bezdek's cluster index (Duda and Hart 1973).

Basic classic cluster approaches like hierarchical clustering, agglomerative clustering or probabilistic clustering are known from mathematical statistics (Schürmann 1996). Another widely applied paradigm is prototype based clustering (Duda and Hart 1973). One basic advantage of prototype based methods like the famous c-Means is the intuitive understanding of their concept to use prototypes as representatives of the clusters (Haykin 1999). Several prototype based methods have been established ranging from statistical approaches (Seo and Obermayer 2003) and graph methods (Luxburg 2007) to neural vector quantizers (Gustafson and Kessel 1979, Gath and Geva 1989, Martinetz et al. 1993, Oja and Lampinen 1994).

Further, one can distinguish between hard and soft variants of cluster algorithms (Bezdek 1981, Dave 1990, Gath and Geva 1989), the most famous of which is the Fuzzy c-Means (Dunn 1973). Frequently, respective approaches belonging to the class of vector quantizers, require the data objects and the prototypes to be embedded in a metric vector space. Popular algorithms, besides the above mentioned c-Means variants, are the Neural Gas (Martinetz et al. 1993), Deterministic Annealed Vector Quantization (Rose et al. 1992), Information Theoretic Clustering (Lehn-Schiøler et al. 2005) or topographic quantizers like Self-Organizing Maps (Kohonen 1995) or Soft Topographic Vector Quantization (Graepel et al. 1997), to name just a few. All of these algorithms use (dis)similarities between objects and prototypes for cluster determination. Usually the dissimilarities are described in terms of Euclidean distances (Kohonen and Somervuo 2002).

Yet in general, any dissimilarity measure fulfilling some specific requirements depending on the clustering algorithm might be appropriate. For example, as mentioned before, INOKUCHI & MIYAMOTO favored divergences (Inokuchi and Miyamoto 2008). ARNONKIJPANICH, in turn, proposed weighting of the input dimensions for unsupervised vector quantization to improve cluster separation using the Neural Gas algorithm (Arnonkijpanich et al. 2010). In section 3.1 these ideas — divergences as dissimilarity measures and weighting of input dimensions — were picked up to incorporate a relevance parameter into the Fuzzy c-Means algorithm. Together with the deployment of divergences as dissimilarity measures, this results in an algorithm usable for the clustering of functional data.

Another class of prototype based cluster algorithms relaxes the assumption of a metric space embedding for the objects to be clustered and the prototypes (Hammer and Hasenfuss 2007, Cottrell et al. 2004, Hofmann and Buhmann 1997). This restriction is replaced by the weaker requirement that only similarities respectively dissimilarities or discrete measures between the data objects are given. Respective algorithms are referred to as median or relational clustering, message passing or graph clustering methods (Frey and Dueck 2007, Seo and Obermayer 2004, Cottrell et al. 2006). Thereby, an underlying metric is assumed for the given dissimilarities in case of relational data clustering, whereas for median clustering this restriction is dropped. These degrees of freedom lead to constraints for the prototypes. For relational data they are assumed as linear combinations of the data. For median clustering this limitation is further reduced to the minimum demand: prototypes have to be data objects.

Fuzzy clustering of relational data based on Fuzzy c-Means was proposed by HATHAWAY ET AL. (Hathaway et al. 1989), inspired by the earlier work of WINDHAM about graded numerical classification of dissimilarity data (Windham 1985). The approach was extended by HATHAWAY AND BEZDEK for more general dissimilarity data, which have to be, however, at least positive, reflexive and symmetric: In this extension the (non-Euclidean) dissimilarity matrix is transformed by the β-spread transform into an Euclidean matrix, such that the clustering is not longer realized on the original data (Hathaway and Bezdek 1994). Other approaches use the Dempster-Shafer theory of belief functions (or evidence theory) to determine a basic belief assignment (or mass function) to each object, such that the degree of conflicts between the masses given to any two objects reflects their dissimilarity (Denoeux and Masson 2004, Masson and Denoeux 2008).

In section 3.2 two fuzzy median clustering algorithms are presented. In section 3.2.1 the Fuzzy c-Means algorithm is extended to work with general dissimilarity data. Based on the idea of median clustering the prototypes are restricted to be data points itself. In this way the work of HATHAWAY & BEZDEK is continued by combining the Fuzzy c-Means with methodology introduced for crisp median clustering (Cottrell et al. 2006).

Another very powerful clustering algorithm based on similarities only is Affinity Propagation as proposed by FREY & DUECK (Frey and Dueck 2007). Yet, contrary to the Median Fuzzy c-Means the conditions of symmetry and completeness of the (dis)similarity matrix are relaxed. In section 3.2.2 a fuzzy interpretation of the algorithm is presented.

The in the following proposed algorithms are investigated by applying them to an artificial and a real world data set. The obtained cluster solutions are evaluated by different cluster validation methods (see also section 2.4).

3.1 Relevance clustering with Fuzzy c-Means

The Fuzzy c-Means algorithm (FCM) is a popular and well known prototype based clustering method. Based on distances – or more general on dissimilarity measures – the prototypes are positioned as close to the cluster centers as possible. Usually all the components of the input vectors are treated equally, independent of their contribution to the clustering. Inspired be the work of ARNONKIJPANICH (Arnonkijpanich et al. 2010), in this section an extension of the FCM algorithm is presented. This variant incorporates a relevance parameter, which models the weighting of the input dimensions and is adapted with each update step, emphasizing those dimensions with less influence on the clustering. E. g. if some of the vector components of several data samples lie close together and are difficult to differentiate, the separability of the clusters is improved, if these dimensions are accentuated. The proposed algorithm is called Relevance Fuzzy c-Means (R-FCM).

This is contrary to supervised relevance learning, where less relevant vector components are identified and neglected, which reduces the number of input dimensions (Hammer and Villmann 2002). To avoid confusion, in the following the term *relevance clustering* will be used to adress unsupervised clustering methods with weighting of input dimensions and *relevance learning* will be reserved for supervised classification with relevance determination.

As mentioned before, commonly, for FCM clustering the Euclidean distance is employed. Yet, the FCM algorithm works with any dissimilarity measure, possibly in different algorithmic realizations. Replacing the Euclidean distance by a divergence, the algorithm can be applied for the clustering of functional data. While for Euclidean vectors the vector dimensions are treated independently, for functional data the vector components are spatially correlated and each vector is a discrete representation of a (continuous) mathematical function, i. e. each input vector $\boldsymbol{v}$ is a representation of the function $v(x)$. Usually, these functional vectors are very high-dimensional like for example spectral data or histograms. Assuming these functions to be positive with finite $\mathfrak{L}_1$-norm, for certain applications the use of a (generalized) divergence, which takes the functional character of the data into account, would be more appropriate to evaluate the dissimilarity than a metric distance measure (Villmann and Haase 2011, Mwebaze et al. 2011).

In this section the FCM algorithm is remodelled to support relevance clustering and it is demonstrated how the distance function of this modified FCM can easily be replaced by non-metric distance measures like divergences. Selected examples demonstrate an improved clustering in terms of separation and compactness and show that depending on the chosen distance measure the shape of the relevance profile varies.

3.1.1 Incorporating a relevance parameter into FCM

The proposed extension of FCM for relevance clustering relies on a scaled distance measure. The d-dimensional relevance parameter, which scales the dimensions of the input vectors, is denoted as $\boldsymbol{\lambda} \in \mathbb{R}^d$. According to HAMMER & VILLMANN (Hammer and Villmann 2002) the constraints $\lambda_k > 0$ and $\sum_{k=1}^{d} \lambda_k = 1$ are valid and $\boldsymbol{\lambda}$ itself has to be subject to optimization. Since the relevance parameter $\boldsymbol{\lambda}$ is a scaling factor and directly affiliated with the data points and prototypes, it has influence on the metric, which now in the mathematical sense reduces to a mere dissimilarity. For example, analogously to eq. (2.78), $\boldsymbol{\lambda}$ can be incorporated into the squared Euclidean distance which results in

$$\begin{aligned} d^{\lambda}_{Euclid}(\boldsymbol{v}, \boldsymbol{w}) &= (\boldsymbol{\lambda} \circ (\boldsymbol{v} - \boldsymbol{w}))^2 \\ &= (\boldsymbol{\lambda} \circ \boldsymbol{v} - \boldsymbol{\lambda} \circ \boldsymbol{w})^2 \end{aligned} \tag{3.1}$$

where $\boldsymbol{\lambda} \circ \boldsymbol{v}$ and $\boldsymbol{\lambda} \circ \boldsymbol{w}$ are Hadamard products, denoting the element-wise multiplication of two same sized vectors. Refer also to section 2.3.1.

Applying the scaled Euclidean distance, the FCM cost function as defined in eq. (2.27) now changes to

$$E_{RFCM} = \sum_{i=1}^{N_V} \sum_{j=1}^{N_P} u_j(\boldsymbol{v}_i)^m \cdot d^{\lambda}_{Euclid}(\boldsymbol{v}_i, \boldsymbol{w}_j). \tag{3.2}$$

Analogously, the update rule for the fuzzy assignments (2.28) now yields

$$u_j(\boldsymbol{v}_i) = \frac{\sqrt[m-1]{d^{\lambda}_{Euclid}(\boldsymbol{v}_i, \boldsymbol{w}_j)^{-1}}}{\sum_{k=1}^{N_P} \sqrt[m-1]{d^{\lambda}_{Euclid}(\boldsymbol{v}_i, \boldsymbol{w}_k)^{-1}}} \tag{3.3}$$

while the prototype update rule (2.29)

$$\boldsymbol{w}_j = \frac{\sum_{i=1}^{N_V} u_j(\boldsymbol{v}_i)^m \cdot \boldsymbol{v}_i}{\sum_{i=1}^{N_V} u_j(\boldsymbol{v}_i)^m}$$

remains unaffected by the modified distance measure.

For the adaption of the relevance parameter $\boldsymbol{\lambda}$ an additional update step has to be added to the FCM algorithm. While the standard FCM consists of two alternating update steps, relevance clustering requires a third update step for the relevance update. The rule therefore can be obtained by stochastic gradient descend learning. The derivative of the cost function E_{RFCM} with respect to component k of the relevance parameter yields

$$\frac{\partial E_{RFCM}}{\partial \lambda_k} = \sum_{i=1}^{N_V} \sum_{j=1}^{N_P} u_j(\boldsymbol{v}_i)^m \frac{\partial d^{\lambda}(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \lambda_k}. \tag{3.4}$$

The update rule for $\boldsymbol{\lambda}$ can now be formulated as

$$\begin{aligned}\Delta\lambda_k &= -\varepsilon\frac{\partial E_{RFCM}}{\partial\lambda_k} \\ &= -\varepsilon\sum_{i=1}^{N_V}\sum_{j=1}^{N_P} u_j(\boldsymbol{v}_i)^m \frac{\partial d^{\lambda}(\boldsymbol{v}_i,\boldsymbol{w}_j)}{\partial\lambda_k}\end{aligned} \tag{3.5}$$

where ε is the learning rate and $1 \leq k \leq d$. During the adaption the constraints for λ_k have to be kept. For the Euclidean distance d^{λ}_{Euclid} update rule (3.5) yields

$$\Delta\lambda_k^{Euclid} = -\frac{\varepsilon}{2}\sum_{i=1}^{N_V}\sum_{j=1}^{N_P} u_j(\boldsymbol{v}_i)^m(\lambda_k v_i^k - \lambda_k w_j^k)(v_i^k - w_j^k) \tag{3.6}$$

Since for the FCM update the metric is assumed to be fixed, the adaption of $\boldsymbol{\lambda}$ has to be performed in an adiabatic manner, to allow the optimization process to follow the drift (Kato 1950). This adiabatic behaviour can be realized by very small learning rates ε.

3.1.2 Divergences as dissimilarity measures

If a divergence $D(\boldsymbol{v}||\boldsymbol{w})$ is used as dissimilarity measure, the FCM cost function (2.27) changes to

$$E^D_{FCM} = \sum_{i=1}^{N_V}\sum_{j=1}^{N_P} u_j(\boldsymbol{v}_i)^m D(\boldsymbol{v}_i||\boldsymbol{w}_j) \tag{3.7}$$

simply by replacing the distance $d(\boldsymbol{v}_i,\boldsymbol{w}_j)$ by the divergence $D(\boldsymbol{v}_i||\boldsymbol{w}_j)$. This replacement is justified in section 2.3.2, where the properties of divergences are compared to the metric axioms and it is concluded, that divergences can be used as dissimilarity measures.

The substition can analogously be transfered to the idea of relevance clustering with FCM. The divergence D^{λ} incorporates the relevance parameter $\boldsymbol{\lambda}$ by weighting the data as well as the prototypes by $\boldsymbol{\lambda}$ applying the Hadamard product:

$$D^{\lambda}(\boldsymbol{v}||\boldsymbol{w}) = D(\boldsymbol{\lambda}\circ\boldsymbol{v}||\boldsymbol{\lambda}\circ\boldsymbol{w}). \tag{3.8}$$

The cost function based on $D^{\lambda}(\boldsymbol{v}||\boldsymbol{w})$ yields

$$E^{\lambda}_{FCM} = \sum_{i=1}^{N_V}\sum_{j=1}^{N_P} u_j(\boldsymbol{v}_i)^m D^{\lambda}(\boldsymbol{v}_i||\boldsymbol{w}_j). \tag{3.9}$$

Algorithm 3.1.1 – Relevance Fuzzy c-Means (R-FCM)

1. Initialize all components of $\boldsymbol{\lambda}$ with $1/d$
2. Initialize the prototypes $\boldsymbol{w}_j$ randomly
3. Calculate the fuzzy cluster assignment u_{ij}
4. Perform FCM learning
   (a) Update fuzzy assignments u_{ij} with fixed prototypes w_j according to eq. (3.3)

$$u_j(\boldsymbol{v}_i) = \frac{\sqrt[m-1]{d^{\lambda}_{Euclid}(\boldsymbol{v}_i, \boldsymbol{w}_j)^{-1}}}{\sum_{k=1}^{N_P} \sqrt[m-1]{d^{\lambda}_{Euclid}(\boldsymbol{v}_i, \boldsymbol{w}_k)^{-1}}}$$

   (b) Update prototypes w_j with fixed fuzzy assignments u_{ij} according to eq. (2.29)

$$\boldsymbol{w}_j = \frac{\sum_{i=1}^{N_V} u_j(\boldsymbol{v}_i)^m \cdot \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j}}{\sum_{i=1}^{N_V} u_j(\boldsymbol{v}_i)^m}$$

   (c) Repeat steps (a) – (c) for a fixed number of iterations, until convergence or manual stop
5. Update relevance parameter $\boldsymbol{\lambda}$ according to (3.5)

$$\Delta\lambda_k = -\varepsilon \sum_{i=1}^{N_V} \sum_{j=1}^{N_P} u_j(\boldsymbol{v}_i)^m \frac{\partial d^{\lambda}(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \lambda_k}$$

6. Repeat steps 4. – 6. for a fixed number of iterations, until convergence or manual stop

Updating the relevance parameter by stochastic gradient descent the derivatives with respect to $\boldsymbol{\lambda}$ have to be calculated. Analogously to (3.4) this derivative yields

$$\frac{\partial E^{\lambda}_{FCM}}{\partial \lambda_k} = \sum_{i=1}^{N_V} \sum_{j=1}^{N_P} u_j(\boldsymbol{v}_i)^m \frac{\partial D^{\lambda}(\boldsymbol{v}_i \parallel \boldsymbol{w}_j)}{\partial \lambda_k}. \tag{3.10}$$

This general function depending on a divergence $D^{\lambda}(\boldsymbol{v} \parallel \boldsymbol{w})$ can easily be adapted to a special divergence, e. g. Kullback–Leibler or Rényi divergence. In section 2.3.2 several divergences from the three divergence families

(i) Bregman-divergences

(ii) Csiszár's f-divergences

(iii) γ-divergences

according to CICHOCKI ET AL. (Cichocki and Amari 2010, Cichocki et al. 2009) were introduced.

To use divergences as distance measures, it is assumed that the parameters $\boldsymbol{v}$ and $\boldsymbol{w}$ are densities, i. e. $v_k, w_k \geq 0$ and $\sum_{k=1}^{d} v_k = 1$ and $\sum_{k=1}^{d} w_k = 1$. By weighting $\boldsymbol{v}$ and $\boldsymbol{w}$ by $\boldsymbol{\lambda}$, $\boldsymbol{\lambda} \circ \boldsymbol{v}$ and $\boldsymbol{\lambda} \circ \boldsymbol{w}$ are no longer densities, but still positive measures. Since the positivity of $\boldsymbol{v}$ can be taken as granted and the positivity of $\boldsymbol{w}$ is assured by (2.29), for this kind of data *generalized* divergences can be used.

In the following their respective derivatives incorporating the relevance parameter $\boldsymbol{\lambda}$ are given. An extensive overview can be found in (Villmann and Haase 2011).

Generalized Kullback–Leibler divergence

For example, consider the generalized Kullback–Leibler divergence $D_{KL}(\boldsymbol{v}||\boldsymbol{w})$, which can be assigned to any divergence class considering some special cases as discussed in section 2.3.2. Taking the relevance $\boldsymbol{\lambda}$ into account, the formula and their respective derivative are given by

$$D_{KL}^{\lambda}(\boldsymbol{v}||\boldsymbol{w}) = \sum_{k=1}^{d} \lambda_k v_k \cdot \mathtt{log}\left(\frac{v_k}{w_k}\right) - \sum_{k=1}^{d} (\lambda_k v_k - \lambda_k w_k) \tag{3.11}$$

$$\frac{\partial D_{KL}^{\lambda}(\boldsymbol{v}||\boldsymbol{w})}{\partial \lambda_k} = v_k \cdot \mathtt{log}\left(\frac{v_k}{w_k}\right) - v_k + w_k \tag{3.12}$$

η-divergence (i)

Analogously, the η-divergence as a representative of the class of Bregman–divergences according to CICHOCKI ET AL. (Cichocki and Amari 2010, Cichocki et al.

2009) can be considered:

$$D_\eta^\lambda(\boldsymbol{v}||\boldsymbol{w}) = \sum_{k=1}^{d}\left(\lambda_k v_k^\eta + (\eta-1)\cdot\lambda_k w_k^\eta - \eta\cdot\lambda_k v_k\cdot(\lambda_k w_k)^{(\eta-1)}\right) \quad (3.13)$$

$$\frac{\partial D_\eta^\lambda(\boldsymbol{v}||\boldsymbol{w})}{\partial\lambda_k} = \lambda_k^{\eta-1}\eta\left(v_k^\eta - w_k^{\eta-1}(v_k\eta+(1-\eta)w_k)\right) \quad (3.14)$$

This divergence is for $\eta = 2$ equivalent with the Euclidean distance and for $\eta \to 1$ it approximates the Kullback–Leibler divergence.

Generalized Rényi divergence (ii)

The respective formulas for the generalized Rényi divergence from the class of Csiszár's f-divergences yield

$$D_R^\lambda(\boldsymbol{v}||\boldsymbol{w}) = \frac{1}{\alpha-1}\,\mathtt{log}\left(\sum_{k=1}^{d}\left[\frac{(\lambda_k v_k)^\alpha}{(\lambda_k w_k)^{(\alpha-1)}} - \alpha\cdot\lambda_k v_k + (\alpha-1)\cdot\lambda_k w_k\right]+1\right) \quad (3.15)$$

$$\frac{\partial D_R^\lambda(\boldsymbol{v}||\boldsymbol{w})}{\partial\lambda_k} = \frac{w_k\cdot((\frac{v_k}{w_k})^\alpha+\alpha-1)-v_k\cdot\alpha}{(\alpha-1)\sum_{l=1}^{d}\left[\lambda_l\cdot\left(w_l(\frac{v_l}{w_l})^\alpha-\alpha\cdot v_l+(\alpha-1)\cdot w_l\right)+1\right]} \quad (3.16)$$

The Rényi divergence approximates for the special case of $\alpha \to 1$ the Kullback–Leibler divergence.

γ-divergence (iii)

The γ-divergence, which is for positive measures very robust with respect to outliers, was proposed by FUJISAWA & EGUCHI (Fujisawa and Eguchi 2008). The respective formulas for relevance learning are:

$$D_\gamma^\lambda(\boldsymbol{v}||\boldsymbol{w}) = \mathtt{log}\left[\frac{\left(\sum_{k=1}^{d}(\lambda_k v_k)^{\gamma+1}\right)^{\frac{1}{\gamma(\gamma+1)}}\cdot\left(\sum_{k=1}^{d}(\lambda_k w_k)^{\gamma+1}\right)^{\frac{1}{\gamma+1}}}{\left(\sum_{k=1}^{d}(\lambda_k v_k)(\lambda_k w_k)^\gamma\right)^{\frac{1}{\gamma}}}\right] \quad (3.17)$$

$$\frac{\partial D_\gamma^\lambda(\boldsymbol{v}||\boldsymbol{w})}{\partial\lambda_k} = \frac{v_k(\lambda_k v_k)^\gamma}{\gamma\sum_{l=1}^{d}(\lambda_l v_l)^2} + \frac{w_k(\lambda_k w_k)^\gamma}{\sum_{l=1}^{d}(\lambda_l w_l)^{\gamma+1}} - \frac{v_k(\gamma+1)(\lambda_k w_k)^\gamma}{\gamma\sum_{l=1}^{d}(\lambda_l v_l)(\lambda_l w_l)^\gamma} \quad (3.18)$$

For $\gamma = 1$ the Cauchy–Schwarz divergence is obtained (Principe et al. 2000).

3.1.3 Proof of concept

Two very basic studies were designed to examine the effect of the adaption of the relevance parameter. Although this is not a formal proof, it will illustrate the concept behind the idea of relevance clustering.

Gaussian distributions

The first artificial data set consists of three two-dimensional Gaussian distributions with varying variance and about 200 data points each. The centers of these distributions are located at $(2, 2)$, $(6, 2)$ and $(4, 6)$. After applying the R-FCM algorithm with three randomly initialized prototypes the relevance parameter changes from $\boldsymbol{\lambda}^{(0)} = [0.50, 0.50]$ to $\boldsymbol{\lambda}^{(250)} = [0.16, 0.84]$ after 250 R-FCM update steps. The difference between $\boldsymbol{\lambda}^{(0)}$ and $\boldsymbol{\lambda}^{(250)}$ implies a compression along the x–axis and a stretch along the y–axis. Fig. 3.1 depicts this changeover by scaling the dimensions one after the other.

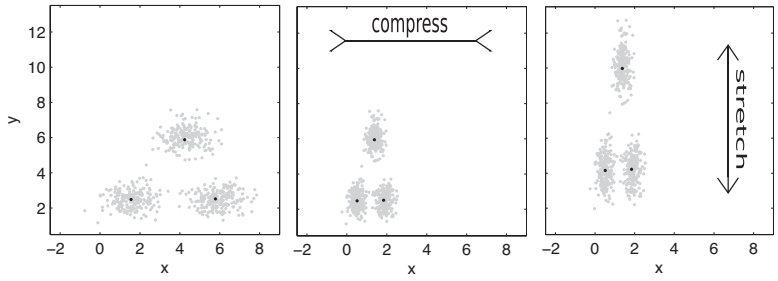

Figure 3.1: *Influence of the relevance parameter on three two–dimensional Gaussian distributions. Weighting dimension 1 compresses the data along the x–axis and weighting dimension 2 stretches the data along the y–axis. These actions result in improved values of selected validation measures.*

In section 2.4.1 different cluster validity indexes based on compactness and separation are described [1]. Applying those to compare the cluster solutions with and without relevance parameter update shows a clear improvement. As depicted in Tab. 3.1

[1]Note, that the actual purpose of these validation measures is to obtain the optimal number of clusters assuming that each cluster is represented by exactly one prototype. Employing these indexes to demonstrate the improvement achieved by adapting the relevance parameter does only make sense if the correct number of prototypes is used.

the values for the three indexes *Partition Entropy*, *Xie–Beni–Index*, and *Fukuyama–Sugeno–Index* decrease, if the relevance parameter is adjusted.

Index	$\boldsymbol{\lambda}^{(0)}$	$\boldsymbol{\lambda}^{(250)}$
PE	1.10	0.47
XB	0.14	0.03
FS	515.91	467.73

Table 3.1: *Value of* Partition Entropy *(PE),* Xie–Beni–Index *(XB), and* Fukuyama–Sugeno–Index *(FS) without relevance parameter adaption (column $\boldsymbol{\lambda}^{(0)}$) and after 250 relevance parameter update steps (column $\boldsymbol{\lambda}^{(250)}$). As expected their values descrease.*

Functionals

The second artificial example consists of only three arbitrary chosen *functionals*, whose values are well separable at the boundaries of the range and lie closer together in the middle. Weighting these functionals by the relevance parameter $\boldsymbol{\lambda} = [0.075, 0.220, 0.220, 0.219, 0.218, 0.048]$ results in the plot depicted in Fig. 3.2. Now the distances between the values at the distinct measure points are evenly separable. Performing a FCM clustering on this kind of weighted data, all dimensions respectively measure points will equally be taken into account and not just those, which show the most distinct difference.

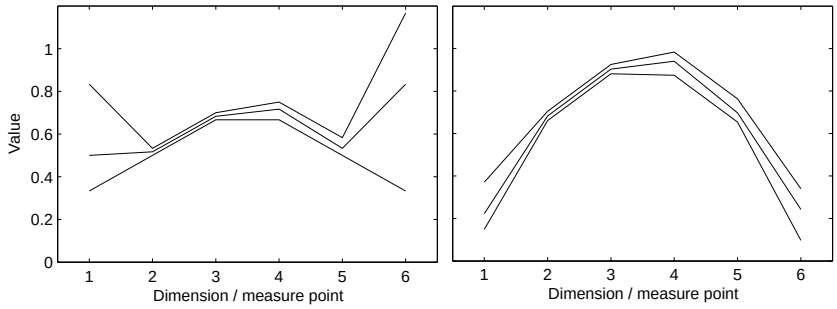

Figure 3.2: *Influence of the relevance parameter $\boldsymbol{\lambda} = [0.075, 0.220, 0.220, 0.219, 0.218, 0.048]$ on the arbitrary artificial functionals (left) resulting in a more evenly spacing of the curves at the measure points (right).*

3.1.4 Real world example – remote sensing data

To illustrate the R-FCM algorithm on a real life data set an example from the biological field is chosen. The Flightline C1 (FLC1) (Landgrebe 2003) remote sensing data was collected by an airborne multi-spectral scanner (MSS) while overflying an agricultural area in the southern part of Tippicanoe County, Indiana. The data consists of 11451 spectra in the range of 0.4 to 2.4 μm. Within this range 12 measures were taken at measure points with varying band widths as indicated in Tab. 3.2. The collected data was assigned to 10 ground cover classes like corn, soybeans, wheat, and others, yet this information was not used for the clustering.

Band #	Wavelength in μm
1	0.40 - 0.44 (0.42)
2	0.44 - 0.46 (0.45)
3	0.46 - 0.48 (0.47)
4	0.48 - 0.50 (0.49)
5	0.50 - 0.52 (0.51)
6	0.52 - 0.55 (0.53)
7	0.55 - 0.58 (0.56)
8	0.58 - 0.62 (0.60)
9	0.62 - 0.66 (0.64)
10	0.66 - 0.72 (0.69)
11	0.72 - 0.80 (0.76)
12	0.80 - 1.00 (0.90)

Table 3.2: *Wavelength ranges of the spectral bands. Values used for the plots are indicated in parenthesis. (Landgrebe 2003)*

For the evaluation of the algorithm a number of distance measures respectively divergences have been selected: Euclidian distance (for means of comparibility), Kullback–Leibler divergence, Rényi divergence, Cauchy–Schwarz divergence, and Gamma divergence (with $\gamma = 0.5$). Randomly 10 prototypes were chosen according to the number of known classes. For means of comparability always the same prototype initializations were used. The initial relevance parameter learning rate was set to 0.001 and per adaption step only 10% of the data samples were considered in order to achieve an adiabatic drift in the optimization process. After executing the known FCM clustering consisting of alternating prototype and assignment update steps, the relevance adaptation step was carried out. Depending on the chosen distance measure a total of 100 to 1000 runs proved to be necessary.

The obtained relevance parameters are plottet in Fig. 3.3. The profiles indicate an emphasis on the lower spectral bands, i. e. if the influence of these bands (input dimensions) is increased during learning, the clustering results will be improved. Remarkable are the relevance profiles for the Kullback–Leibler and the Rényi divergence, since they are almost identical.

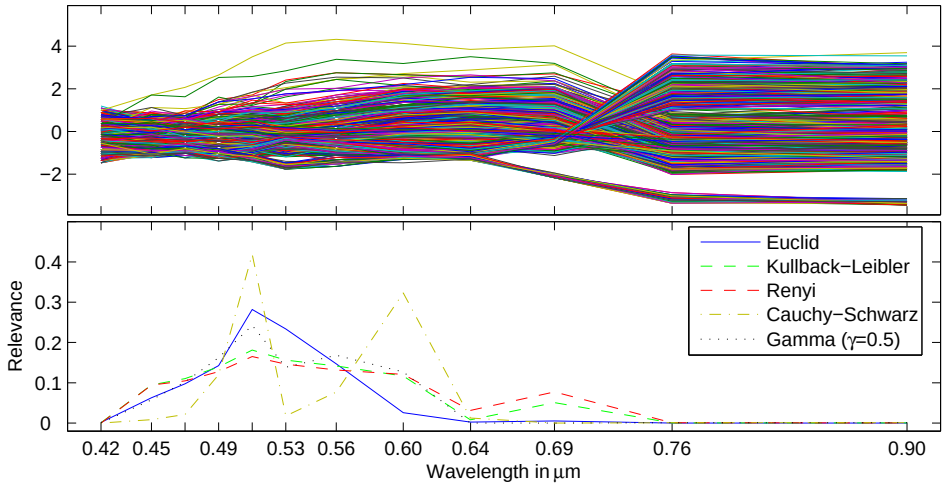

Figure 3.3: *Spectra of the FLC1 data set (upper part) and relevance profiles (lower) obtained for selected distance measures (lower part).*

After completed clustering and the adaption of the relevance parameter the *Partition Entropy* (Bezdek 1974b, Bezdek 1974a), the *Fukuyama–Sugeno–Index* (Fukuyama and Sugeno 1989), and the *Xie–Beni–Index* (Xie and Beni 1991, Pal and Bezdek 1995) were calculated. These cluster validity indexes are described in detail in section 2.4.1. All three measures show the expected behavior, i. e. they decrease after the successful adaption of the relevance parameter indicating an improved clustering in terms of compactness and separation. Detailed results listing the index values with and without relevance adaption can be found in Tab. 3.3.

The impact of the relevance parameter on the data is depicted in Fig. 3.4. The upper part of the figure is a plot of the relvance obtained by R-FCM clustering using the Rényi divergence. The lower part shows the spectral response lines weighted by the relevance. This view of the data reveals details in the structure of the lower bands, which are not distinguishible in the plot of the original data. Plots of the weighted spectral bands obtained for the other above mentioned divergences look similar.

Idx	Euclid	Rényi	Kull.-Leib.	γ-Div.(γ=0.5)	γ-Div.(γ=1.0) Cauchy-Schw.
PE	2.3026	2.3026	2.3026	2.3026	2.3026
	1.6516	1.4923	1.5492	1.4905	0.6474
FS	0.3392	255.0732	47.8509	464.2801	289.6427
	0.0355	4.1854	1.8236	5.7332	8.6354
XB	0.0107	0.0195	0.0126	0.0224	0.0110
	0.0054	0.0052	0.0052	0.0159	6.70e-04

Table 3.3: *Validity index measures for the Remote Sensing FLC1 data set without (1st entry) and with (2nd entry) relevance parameter adaptation. A lower value indicates a better clustering in terms of separation and compactness.*

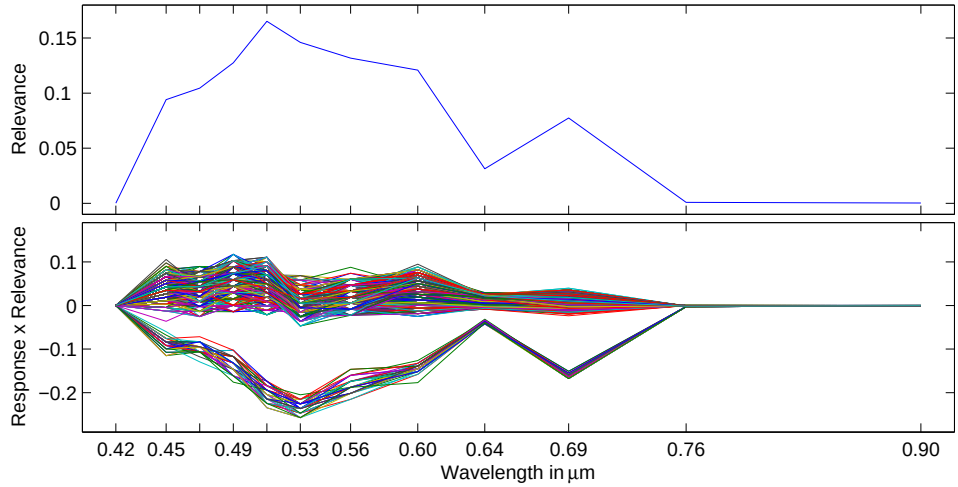

Figure 3.4: *Relevance profiles obtained using Renyi divergence and scaled spectra of the FLC1 data set.*

In Fig. 3.5 the prototypes are plotted on the dimmed response lines of the original unweighted data. It can be observed that these prototypes are *well spread* and are covering the complete range of the values (excluding some outliers). Selecting randomly four prototypes and plotting them together with the original data weighted according to their assigment to the respective prototype shows an *intuitively good clustering*.

Plotting the clusters as 2D colored maps (see Fig. 3.8) and comparing these plots

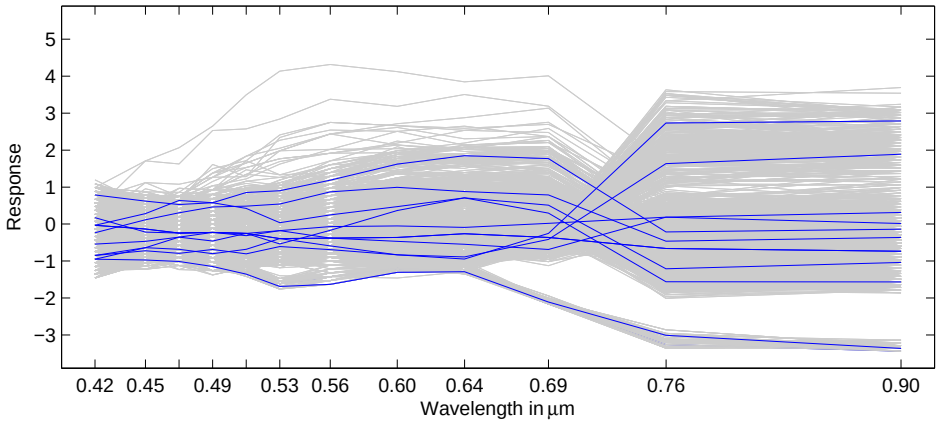

Figure 3.5: *Prototypes (black) obtained for the FLC1 data set (light gray) using Renyi divergence.*

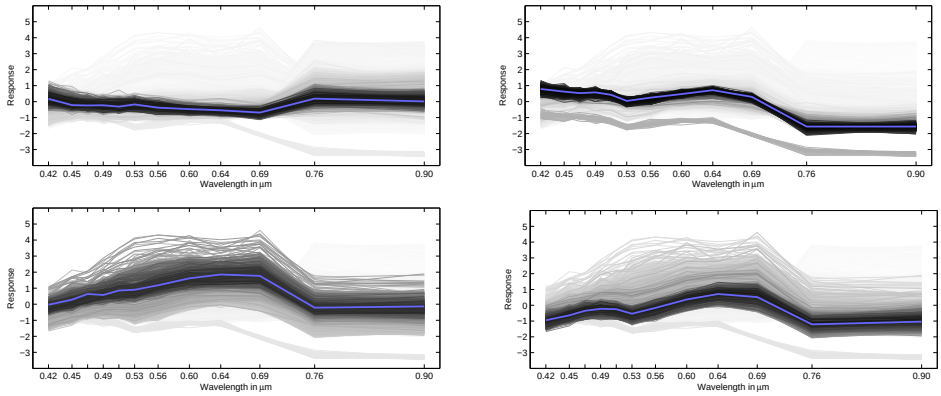

Figure 3.6: *Four prototypes obtained for the FLC1 data and their receptive fields (spectral lines dimmed according to their assignment).*

with the aerial view and the original labeled data set 3.7 shows significant agreements between the cluster results and the class labels of different agricultural fields, i.e. the ten extracted clusters correspond with the ten known classes. Exemplarily the plots for the clusters showing significant agreements with the classes for "wheat", "oats", and "bare soil" are depicted. The intensity of the pixel colors indicates the assignment strength of each pixel to the specific cluster.

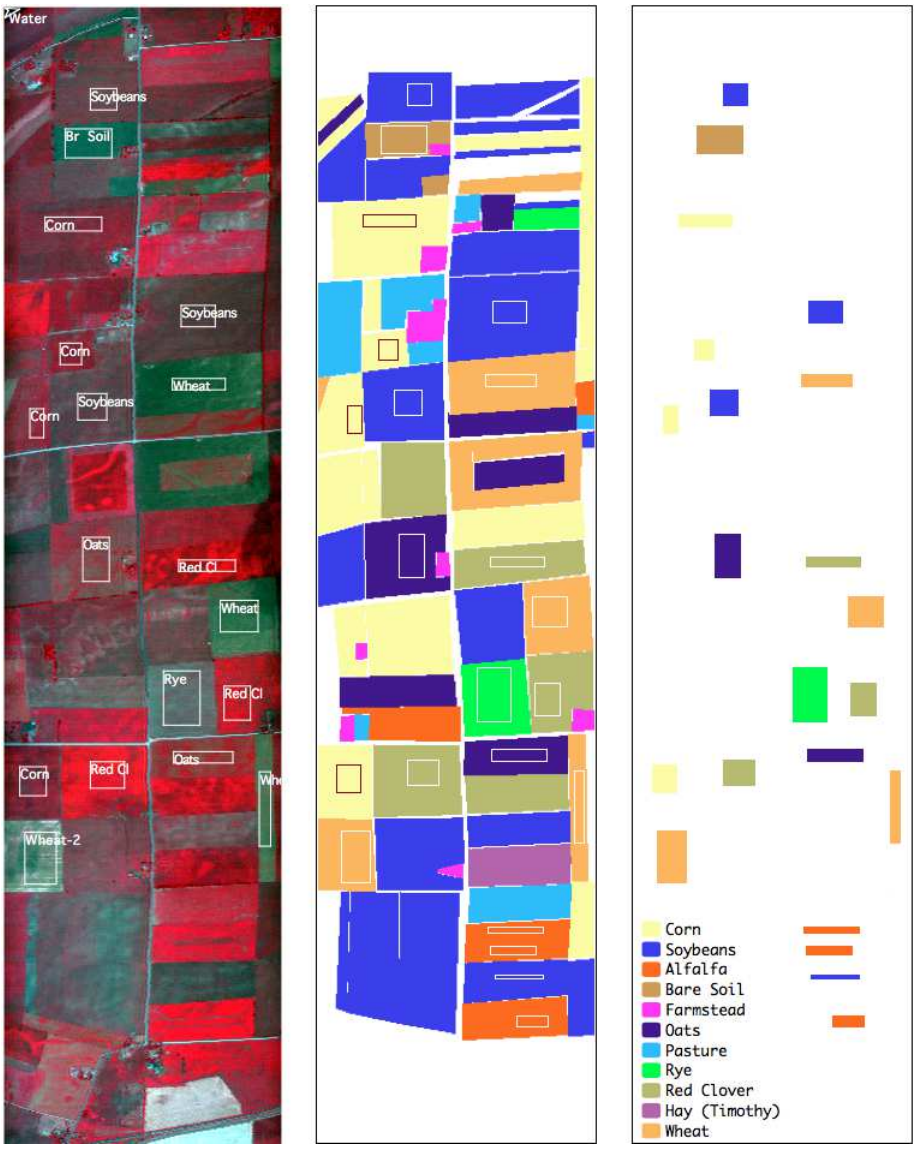

Figure 3.7: *FLC1 data set.* left: *aerial view with marked training areas.* center: *colored "ground truth" map.* right: *extracted training sections. (Landgrebe 2003)*

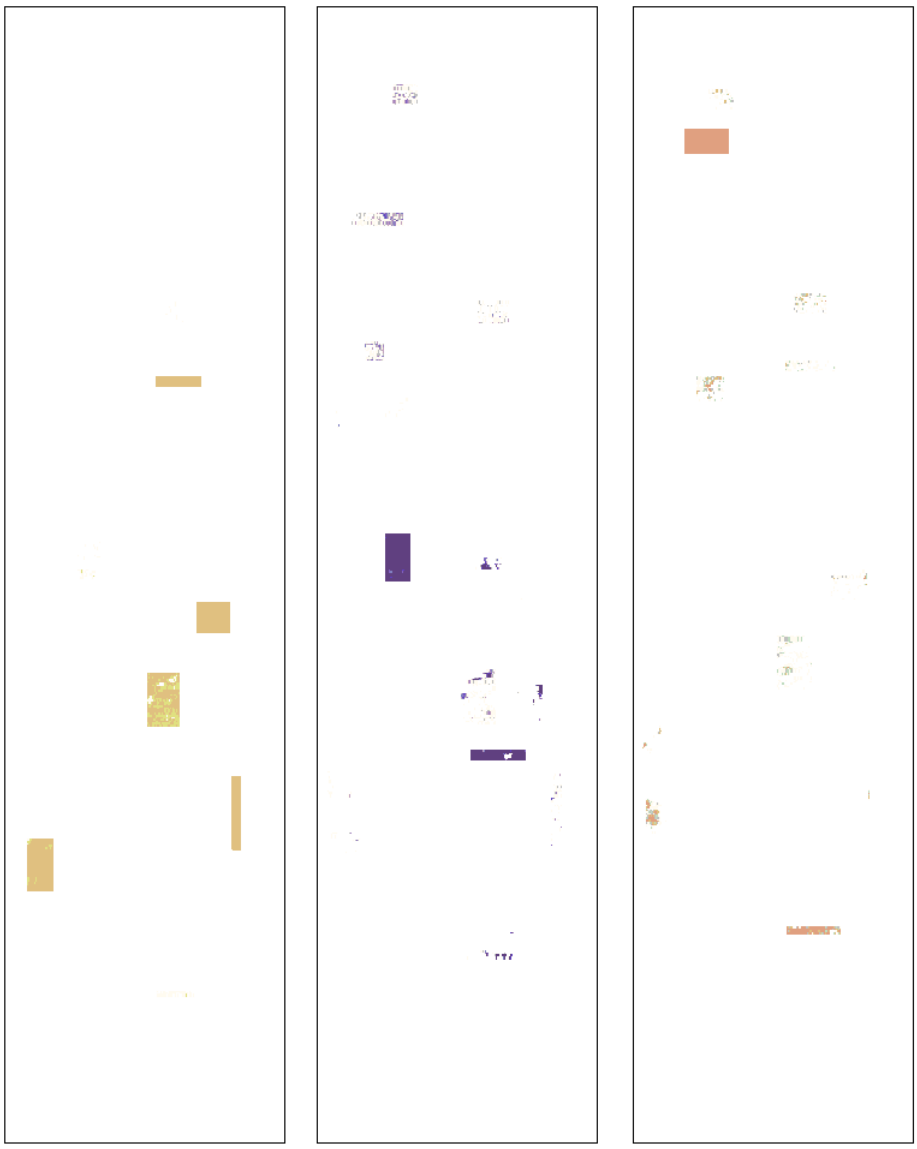

Figure 3.8: *Three selected clusters out of a total of 10 (corresponding to the 10 prototypes). Each cluster is presented separately. The intensity of the colors indicates the strength of the fuzzy assignments for each single data point to the respective cluster. Based on the agreement with the training areas (see Fig. 3.7) and their class labels the clusters can be matched with the classes "wheat", "oats", and "bare soil".*

3.1.5 Conclusion

The FCM algorithm is a widely known clustering method. Commonly it is used with the Euclidean distance as dissimilarity measure. In this chapter it is demonstrated, that divergences like Rényi divergence, Kullback–Leibler divergence, and γ-divergence might be used as well. These are especially applicable for functional data. Appropriate update rules for the protoypes are derived.

Further, it is demonstrated how the relevance parameter can be learned. This implies, that the metric is no longer fixed and can be modulated to improve the representation of the data. This way more accurate cluster solutions in terms of compactness and separation are obtained. The improved algorithm is called R-FCM, where the *R* stands for *relevance*.

The performance of the improved method is demonstrated on a real life data set.

3.2 Fuzzy median clustering

Median clustering algorithms group data based on similarities respectively dissimilarities between the objects, which are not required to be embedded in the Euclidean space. Merely the similarities between the objects, obtained by some kind of distance measure, have to be known. Since the resulting prototypes cannot be positioned inbetween the data objects, they are restricted to be data points by themselves.

In the first part of this section the Median Fuzzy c-Means (M-FCM) is derived by merging the standard Fuzzy c-Means (FCM) (Dunn 1973, Bezdek 1980) and the Median c-Means (M-CM) (Bezdek 1981), see also sections 2.1.5 and 2.1.6 respectively. The dissimilarities have to be supplied in a matrix, which is required to be symmetric and complete. The resulting M-FCM allows fuzzy assignments of the objects to the cluster prototypes.

Another very powerfulf clustering algorithm is Affinity Propagation (AP) proposed by FREY & DUECK (Frey and Dueck 2007), which is a message passing clustering algorithm based on the max–sum–algorithm optimization for factor graphs (Pearl 1988), see also section 2.1.7. Contrary to the Median Fuzzy c-Means (M-FCM) the conditions of similarity and completeness of the (dis)similarity matrix are relaxed. In the second part of this section a probabilistic interpretation of the algorithm, which allows to derive fuzzy clusterings, is provided. In the following this AP variant is referred to as Fuzzy Affinity Propagation (FAP).

The last two subsections provide two examples — an artificial and a real life example — to illustrate the performance of the proposed algorithms.

3.2.1 Median Fuzzy c-Means

In complete analogy to FCM and M-CM (refer to sections 2.1.5 and 2.1.6) the Median Fuzzy c-Means (M-FCM) algorithm performs a two-step iteration scheme of prototype and assignment update to minimize the cost function

$$E_{M\_FCM} = \frac{1}{2} \sum_{j=1}^{N_P} \sum_{i=1}^{N_V} u_j(\boldsymbol{v}_i)^m \cdot d(\boldsymbol{v}_i, \boldsymbol{w}_j) \tag{3.19}$$

which is the same as for FCM (2.27) apart from the fact that now the distances $d(\boldsymbol{v}_i, \boldsymbol{w}_j)$ may be arbitrary dissimilarities as for M-CM. Since these dissimilarities might violate the triangle inequality, the prototypes are restricted to be data points by themselves as it is known from the M-CM. Therefore $d(\boldsymbol{v}_i, \boldsymbol{w}_j) = d(\boldsymbol{v}_i, \boldsymbol{v}_{J(j)})$, where $J(j)$ complies to the mapping introduced in eq. (2.34).

Analogously to most c-Means variants the update of the prototypes and assignments is performed according to an alternating optimization scheme. Note, that the assigment update (3.20) is structurally identical with the FCM assignment update (2.28) and obtained in complete analogy to the derivation described in (Bezdek 1981). For the prototype update it has to be paid attention to the fact, that the prototypes are bound to be data points by themselves. Therefore, the update ist obtained according to that of M-CM (2.38).

Proof of convergence

The convergence of the M-FCM algorithm can be shown using the same arguments as for the convergence proof of median neural gas (Cottrell et al. 2006): Take the assignments $u_j(v_i)$ as functions of the set W of all prototypes $u_j(W)$ and assume that for given W the determination of the assignments $u_j(v_i)$ is unique (may be after definition of ordering in case of ties). Because of the discrete nature of the objects the assignments are a subset of some discrete set K. If the assignments are fixed, choice (2.38) is optimum at least for the continuous case except zero-sets and for the median variant we can assume uniqueness by introducing an order. Now we define the function

$$Q(W', W) = \sum_{j=1}^{N_P} \sum_{i=1}^{N_V} f_1(u_j(W)) \cdot f_2^{ij}(W') \tag{3.20}$$

with $f_1(u_j(W)) = (u_j(W))^m$ and $f_2^{ij}(W') = d(v_i, w_j)$. Hence, $E_{M-FCM}(W) = Q(W, W)$. Assume for given prototypes W, new prototypes W' are derived by the above rule (2.38) based on the assignments $u_j(W)$. Then $E_{M-FCM}(W') = Q(W', W')$. On the one hand side $Q(W', W') \leq Q(W', W)$ is valid, because $u_j(W')$

Algorithm 3.2.1 – Median Fuzzy c-Means

1. Select random data points $\boldsymbol{v}_{J(j)}$ as initial prototypes $\boldsymbol{w}_j$
2. Update fuzzy cluster assignments u_{ij} with fixed prototypes $\boldsymbol{w}_j$ according to eq. (2.28) as known from FCM

$$u_j(v_i) \quad = \quad \frac{\sqrt[m-1]{d(\boldsymbol{v}_i, \boldsymbol{v}_{J(j)})^{-1}}}{\sum_{k=1}^{N_P} \sqrt[m-1]{d(\boldsymbol{v}_i, \boldsymbol{v}_{J(k)})^{-1}}}$$

3. Update prototypes w_j with fixed fuzzy assignments u_{ij} according to eq. (2.38) as known from M–CM

$$\boldsymbol{w}_j = \boldsymbol{v}_l \quad \text{with} \quad l = \underset{l'}{\operatorname{argmin}} \left(\sum_{i=1}^{N_V} u_j(\boldsymbol{v}_i)^m \cdot d(\boldsymbol{v}_i, \boldsymbol{v}_{l'}) \right)$$

4. Repeat steps 2. and 3. until convergence or manual stop

are assumed to be optimum in E_{M-FCM} for W'. This assumption is valid since they are determined according to the derivatives of the recpective Lagrangian of the cost function, which is the same as for FCM given by (2.30), despite the restrictive setting for the prototypes. On the other hand, $Q(W', W') \leq Q(W, W)$ holds because W' are optimum choices for given $u_j(W)$. Thus, $E_{M-FCM}(W') - E_{M-FCM}(W) \leq 0$ and, therefore, E_{M-FCM} is decreased at each step of the algorithm. Since there is a finite number of different values $u_j(v_i)$ the M-FCM algorithm converges in a finite number of steps.

3.2.2 Fuzzy interpretation of Affinity Propagation

Affinity Propagation – introduced by FREY & DUECK (Frey and Dueck 2007), see also section 2.1.7, – can be seen as an exemplar-dependent probability model, where the given similarities $s(i, k)$ between N_V data points $v_i, v_k \in N_V$ (potential exemplars) are identified as log-likelihoods of the probability that the data points assume each other as prototypes. More specific, the similaritites $s(i, k)$ between the data points v_i and v_k may be interpreted as log-likelihoods. Therefore, the cost function

given in (2.42)

$$S(I) = \sum_{i=1}^{N_V} s\left(i, I(i)\right) + \sum_{j=1}^{N_P} \delta_j(I)$$

where the mapping function $I(j) : N_V \rightarrow N_V$ defines the prototypes for each data point and with the penalty function (2.43)

$$\delta_j(I) = \begin{cases} -\infty & \text{if } \exists j,k \quad I(j) \neq j, I(k) = j \\ 0 & \text{otherwise} \end{cases}$$

can be replaced by a cost function based on on the log-probabilities

$$S(I) = \log\left(\prod_{i=1}^{N_V} P(i, I(i)) \cdot P(I)\right) \tag{3.21}$$

with $P(i, I(i))$ as the probability that $I(i)$ is the prototype for v_i and $P(I)$ is the probability that this assignment is valid. Note, that normalization does not affect the solution.

To obtain a fuzzy interpretation of the clustering, assume that an exemplar set $I_P \subset N_V$ has emerged, i. e. the prototypes are known. For each data point $v_i, v_k \in I_P$ a cluster probability $P(i,k) = 0$ iff $i \neq k$ and $P(i,i) = 1$ can be defined. According to FREY & DUECK (Frey and Dueck 2007) the responsibilities $r(i,k)$ measuring the degree, to which data point $v_k \in I_P$ is suitable to be the prototype for data point v_i, can be taken as log-probability ratios. To ensure a probability description of cluster assignments the normalized responsibilities for non-prototypes are introduced as

$$\widehat{r}(i,k) = C \frac{r(i,k) - \max_{i|v_i \notin I_P}\{r(i,k)\}}{\max_{i|v_i \notin I_P}\{r(i,k)\} - \min_{i|v_i \notin I_P}\{r(i,k)\}} \tag{3.22}$$

If the normalization constant C is chosen appropriately based on the variance of $r(i,k)$, the probabilities for the mapping of the data points v_i onto a cluster prototype v_j can now be defined by

$$P(i,j) = e^{\widehat{r}(i,j)}, \quad P(i,j) \in [0,1]. \tag{3.23}$$

This definition of $P(i,j)$ is compliant with the log-ratio interpretation of the responsibilities $r(i,j)$ and can be interpreted as fuzzy cluster assignments. A probabilistic clustering can be obtained by subsequent normalization of $P(i,j)$.

3.2.3 Artificial example – overlapping Gaussian distributions

The data sets for this first experiment are created from five overlapping Gaussian distributions. These data sets are used to demonstrate the performance of the Median Fuzzy c-Means (M-FCM) and the fuzzy interpretation of Affinity Propagation (FAP) in comparison to other algorithms: Median c-Means (M-CM), Median Neural Gas (MNG) and standard Fuzzy c-Means (FCM). Two different settings were used: in the first setting there is a substantial overlap whereas for the second run the Gaussians are separated more clearly, see Fig. 3.2.3 and 3.11.

The centers of the Gaussians are located at $(5, 4)$, $(11, 4)$, $(13, 10)$, $(8, 10)$, and $(3, 10)$. Each distribution consists of 100 data points and the variance within the respective clusters is set to 1.5 for the overlapping data set and to 1.0 for the well separated data set.

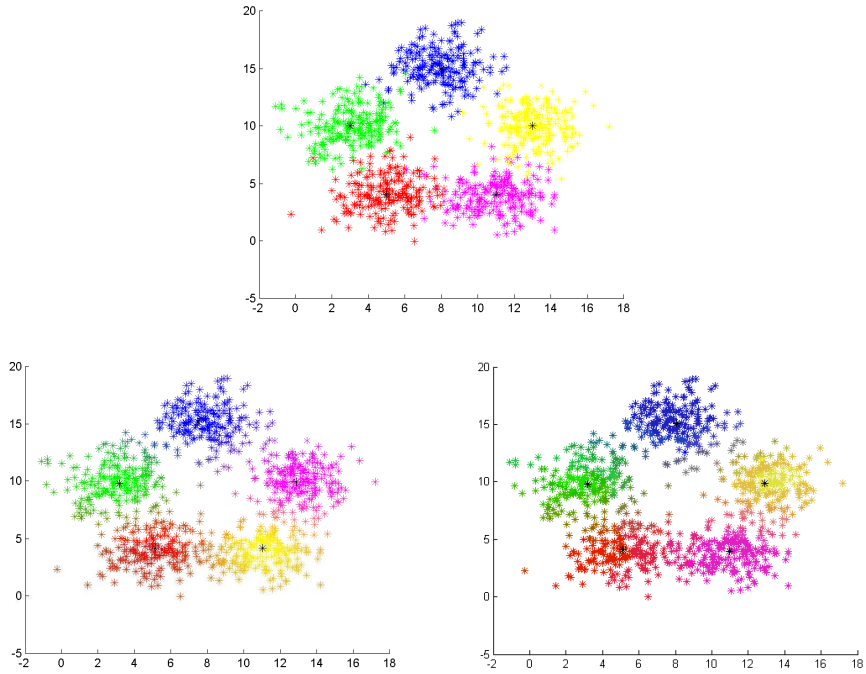

Figure 3.9: *Artificial data set consisting of 5 overlapping Gaussian distributions with 100 data points each and a variance of 1.5. On top the original data set used for the experiments and below the fuzzy clustering obtained by M-FCM (left) and FAP (right).*

For the M-FCM five prototypes were randomly initialized and the fuzziness parameter m was set to $m = 2$. Several runs with different prototype initializations

have been performed.

The initialization of the preferences for the FAP, which coincide with the initial values for $s(i,i)$ (see section 2.1.7), had been chosen appropriatly, since these preferences indirectly influence the number of obtained clusters and it is important, that the cluster count for all algorithms matches due to the Kappa value (see section 2.4.3) which was applied to measure the accordance between the cluster distributions of two or more algorithms and which can only be used for solutions where the number of clusters is identical.

For the overlapping data set the results of various M-FCM runs varied only slightly. Therefore the result of an arbitrary run was selected by chance to evaluate the performance of M-FCM. For FAP only one relevant run could be perfomed. Runs conducted after changing the initial settings, i. e. the preferences, resulted in different cluster numbers and had to be discarded due to evaluation requirements.

Additionally, standard FCM based on the original vectorial data, M-CM and MNG were applied. The latter two are crisp clustering algorithms and MNG is known to be a robust variant of M-CM (Cottrell et al. 2006).

The resulting prototypes of all the applied algorithms are very close , see Fig. 3.10. The fuzzy cluster assignments for M-FCM and FAP are depicted in Fig. 3.2.3. The agreement between the cluster solutions is also reflected by the pairwise comparisons in terms of fuzzy Cohen's Kappa $\hat{\kappa}_C$ (refer to section 2.4.3) as given in Table 3.4.

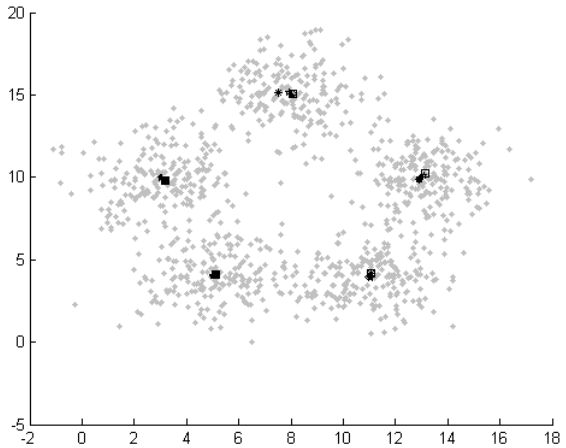

Figure 3.10: *Prototypes for the overlapping data set: $*$ M-FCM, $\diamond$ FAP, $\square$ M-CM, $\star$ FCM, $\circ$ MNG. The positions of the prototypes are almost identical.*

	M-FCM	FAP	FCM	M-CM	MNG
M-FCM	—	0.84	0.81	0.75	0.73
FAP	0.84	—	0.70	0.64	0.64
FCM	0.81	0.70	—	0.93	0.93
M-CM	0.75	0.64	0.93	—	1.00
MNG	0.73	0.64	0.93	1.00	—

Table 3.4: *Fuzzy Cohen's Kappa $\hat{\kappa}_C$ obtained by comparing different cluster algorithms for the overlapping data set with each other. Moderate to substantial agreements according to Tab. 2.1 can be observed between fuzzy algorithms.*

It can be observed that FAP and M-FCM show small systematic deviations from the other algorithms due to their weak assumptions and their great flexibility for assignments (fuzzy). Otherwise, exact identical cluster solution are found by MNG and M-CM which yields a Cohen's Kappa value of $\hat{\kappa}_C = 1.00$. The multiple comparison of the fuzzy cluster algorithms M-FCM, FAP, and FCM gives a Fleiss' Kappa $\hat{\kappa}_F = 0.74$ (see also section 2.4.3), which also refers to a high overall agreement.

Yet, it should be mentioned at this point that the comparison of agreements does not tell anything about accuracy. It only judges the agreement of different methods. However, because the three other methods — M-CM, MNG, and FCM — are already evaluated to deliver good clustering results for several applications in different areas, it can be concluded that M-FCM and FAP will probably do well, too. Thereby, again, it has to be taken into account that a comparison of crisp and fuzzy methods using $\hat{\kappa}_C$ or/and $\hat{\kappa}_F$ necessarily results in lower values than the comparison among crisp or fuzzy methods separately, because the degree of freedom of the assignments is different.

If the Gaussians are separated more clearly as depicted in Fig. 3.11, the assignments of the clusters become crisper and M-FCM and FAP tend to get stuck in local minima depending on the initialization. This property is also known from M-CM and can be reduced by the introduction of neighborhood cooperativeness between the prototypes as it is realized in the MNG approach. But the effect is not vanishing (Cottrell et al. 2006). Hence, this behavior can be seen as a consequence of the median restriction of the prototypes and should not be addressed to the introduced fuzziness in M-CM and FAP.

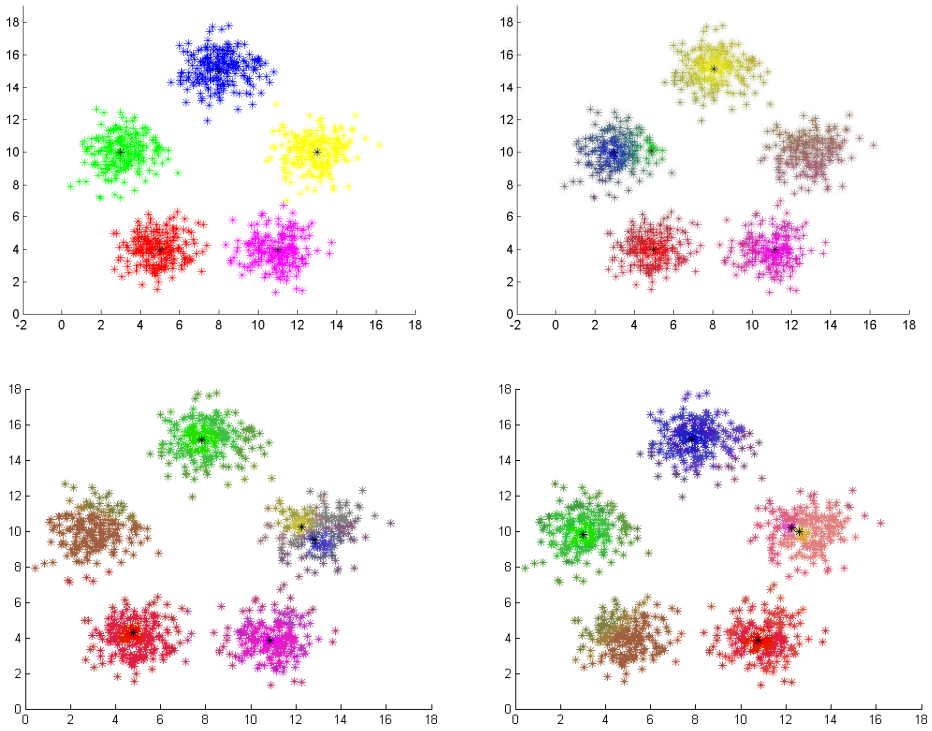

Figure 3.11: *Artificial data set consisting of 5 well separated Gaussian distributions with 100 data points each and a variance of* 1.0*. Top left: original data set used for the experiments. Top right:s cluster solution obtained by FAP. Bottom: two different fuzzy clusterings obtained by M-FCM for different prototype initializations.*

3.2.4 Real world example – psychological therapy transcripts

The real life data set is from the medical field and consists of a series of psychological therapy session transcripts. These text documents contain the exact wording of the dialogs between the psychotherapist and the patient together with some additional annotations concerning the mood of the patient. Each session lasted about 45 minutes and the data set covers 35 sessions of one and the same patient and therapist.

To cluster the data the Normalized Compression Distance NCD (2.87) (see section 2.3.4) based on the Kolmogorov complexity was used to generate a dissimilarity matrix D^{Dis}. Yet since FAP requires a similarity matrix, D^{Dis} was transformed to

a similarity matrix D^{Sim} according to $D^{Sim}_{x,y} = 1 - D^{Dis}_{x,y}$. The resulting matrix is complete, but due to the nature of NCD not symmetric. To be able to use M-FCM for cross-validation, the matrix elements describing the same pair were replaced by their means, i. e. $(D_{x,y} + D_{y,x})/2$. For further considerations only the symmetrized matrixes were used, see also Fig. 3.12.

A psychological therapy is often a two phase transition process. In order to capture this process in the clustering, two prototypes were provided for M-FCM. For FAP the initial preferences $s(i,i)$ have to be set to $s(i,i) = 2 \cdot min(s(i,k))$ to prompt two exemplars to emerge. For means of comparison also a crisp clustering is generated using MCM. The respective resulting clusters are depicted in Fig. 3.12, where the cluster assignments are represented by a color transition from magenta (phase 1) to cyan (phase 2).

It can clearly by observed that there is a smooth phase transition around session 17 indicating a break through in the therapy. This observation is also manifested by considering the NCD dissimilarity matrix and by clinical findings (Villmann et al. 2008, Villmann et al. 2010).

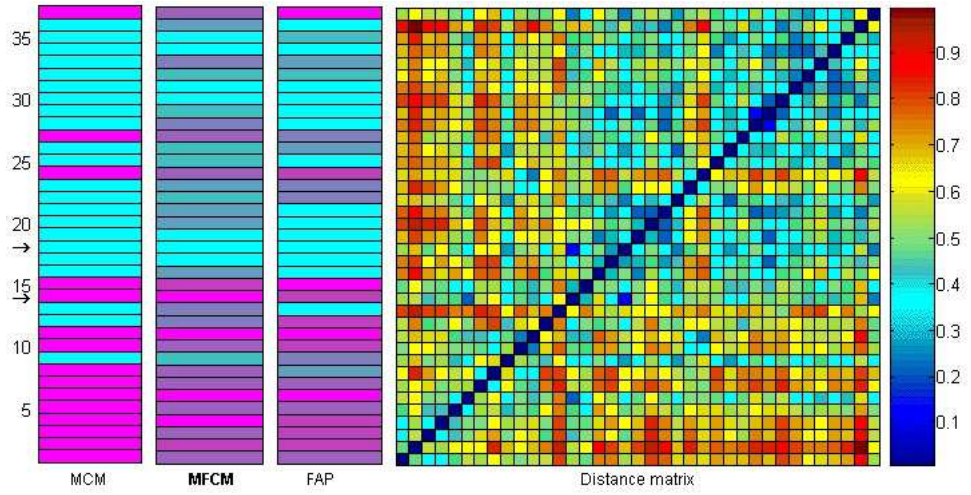

Figure 3.12: *Visualization of the clustering results for the psychotherapy session verbatim protocols. Left: the cluster results for MCM, M-FCM and FAP. The fuzzy assignment is coded in a blue−magenta color range. Pure color indicates a clear cluster decision according to the reference whereas color shades stand for uncertain decisions. A nice agreement can be observed. The sessions acting as prototypes for M-FCM are indicated by arrows on the left. The transition from phase 1 (magenta) to phase 2 (blue) around session 17 can be clearly detected. Right: dissimilarity matrix. The vertical as well as the horizontal ordering is according to the session number.*

Additionally to the visual inspection also the Kappa value as described in section 2.4.3 was calculated. Fuzzy Fleiss' Kappa $\widehat{\kappa}_F$, measuring the inter-rater agreement of more than two classifiers, yields $\widehat{\kappa}_F = 0.25234$ indicating a fair agreement between the cluster solutions obtained by M-CM, M-FCM, and FAP. This relative low overall agreement is mainly attributed to the great deviations of the fuzzy assignments in comparison to the crisp decision caused by M-CM. The pairwise agreement by Cohen's Kappa $\widehat{\kappa}_C$ between the two fuzzy approaches FAP and M-FCM is $\widehat{\kappa}_C = 0.44918$ and can be interpreted as a moderate agreement, which is rather high in the field of statistical comparison of verbal items in psychotherapy (Albani et al. 2002).

The surprising side-effect of clustering narrative data of psychotherapy sessions is that obviously the psychotherapy verbatim protocols contain much structural text information about the behavior or the behavioral disparity in different stages of the therapy. As mentioned above, the soft phase transition around session 17 occurs in concordance with clinical findings, where several psycho-physiological parameters were investigated. Therefore different measures like the patients blood pressure, heart rate, and skin resistance were taken during the therapy session. The analysis of those parameters yields a similar soft phase transition, which is according to (Villmann et al. 2003) triggered by the therapeutic interventions. Yet, those complex investigations are very time consuming. Using therapy narratives offers a cheap yet less accurate alternative which could be considered as a first step to analyze the therapeutic process.

3.2.5 Conclusion

In the previous section two prototype based non-hierarchical fuzzy median clustering algorithms are introduced. The first is the Median Fuzzy c-Means, which is a combination of the known Fuzzy c-Means (Dunn 1973), see section 2.1.5, and the Median c-Means (Bezdek 1981), see section 2.1.6. The other algorithm is an extension of Affinity Propagation (Frey and Dueck 2007), see section 2.1.7, where a probabilistic interpretation is used to remodel the calculation of the responsibilities. In this modified form they can be used for fuzzy membership assignments of the data to the cluster centers respectively prototypes or exemplars.

Both algorithms have in common, that they can be applied to non-vectorial data since they merely rely on similarities respectively dissimilarities between the data samples. They are appropriate for many clustering tasks where non-metric data objects like texts, questionnaire lists, etc. have to be grouped as it is frequently the case in social sciences, psychology or medical applications, where any metric assumptions about the similarities cannot be assured. Thereby, the fuzzy assignments

of the data to the cluster prototypes offer greater flexibility in comparison to crisp median clustering approaches like M-CM or standard AP while keeping the easy interpretability of prototype based clustering.

The (dis)similiarites between the data samples have to be provided, whereby these should follow the common sense understanding of *(dis)similarity* (Pękalska and Duin 2005). Yet the requirements for the (dis)similarity matrixes vary: While the dissimilarity matrix for M-FCM has to be complete and symmetric, there are no restrictions put on the simmilarity matrix for FAP. The prototypes of both algorithms are restricted to be data points themselves, as it is common for median clustering methods.

The M-FCM algorithm tends to get stuck in local minima such that several runs have to be performed to obtain good performance. This behavior is known also from the other median clustering approaches and can be addressed to the discrete behavior of prototype adaptation, i. e. the change of a prototype connotes in practice that another data point serves as prototype and means a drastic change in the configuration influencing heavily the cost function value. This could cause problems for sparse data sets, if the median variant does not offer sufficient flexibility for prototype adaptation. This tendency could possibly be reduced, if M-FCM would be further combined with the more stable variant of M-CM, the Median Neural Gas, which has not been done yet and will addressed during further investigations. Another future link could be to integrate label information for semi-supervised clustering as it is also known for Median Neural Gas (Hammer et al. 2009).

In FAP the number of the clusters cannot be set directly. Instead, the optimal number of exemplares emerges without further user interaction. To obtain a solution with a certain number of clusters, for example to compare the solution with a cluster solution obtained by another algorithm, FAP has to be run several times with varying settings for the preferences $s(i, i)$ until the requested partitioning is reached.

The main disadvantages for both algorithms is the necessity to store and to handle the complete dissimilarity matrix, which is with quadratic cost depending on the number of data. Yet, this is not specific to M-FCM and FAP but rather to all median approaches and could limit the applicability of the algorithms. Fortunately, patch variants are proposed to solve this disadvantage for MNG (Alex et al. 2009) and Relational FCM (Bezdek et al. 2006), which can easily be adopted for M-FCM.

Chapter based on:

Tina Geweniger and Thomas Villmann – *"Extending FSNPC to handle data points with fuzzy class assignments,"* in M. Verleysen (ed.), Proc. of European Symposium on Artificial Neural Networks (ESANN 2010), d-side publications, Evere, Belgium, 2010.

Tina Geweniger, Petra Schneider, Frank-Michael Schleif, Michael Biehl, and Thomas Villmann – *"Extending RSLVQ to handle data points with uncertain class assignments,"* in Machine Learning Reports 2/2009, ISSN:1865-3960, 2009.

Petra Schneider, Tina Geweniger, Frank-Michael Schleif, Michael Biehl, and Thomas Villmann – *"Multivariate class labeling in Robust Soft LVQ,"* in M. Verleysen (ed.), 19th European Symposium on Artificial Neural Networks (ESANN 2011), d-side publications, Evere, Belgium, 2011.

Thomas Villmann, Barbara Hammer, Frank-Michael Schleif, Tina Geweniger, and Wieland Herrmann – *"Fuzzy classification by fuzzy labeled neural gas,"* Neural Networks, vol. 19, no. 6-7, pp. 772–779, 2006.

Thomas Villmann, Udo Seiffert, Frank-Michael Schleif, Cornelia Brüß, Tina Geweniger, and Barbara Hammer – *"Fuzzy labeled self-organizing map with label-adjusted prototypes,"* in Proc. of Conf. Artificial Neural Networks in Pattern Recognition (ANNPR 2006), Springer, Ulm, Germany, 2006.

Chapter 4

Fuzzy classification

Abstract

In this chapter several known classifiers are adapted to work with fuzzy labeled data. An enhancement of the Soft Nearest Prototype Classifier introducing fuzzy labeled prototypes is extended to handle uncertain data class assignments. Further, an advanced LVQ method, namely Robust Soft LVQ which is based on parametrized probability estimates for class and data distributions, is modified to handle fuzzy data labels and prototype assignments. For both the respective update rules are derived, for the RSLVQ variant also considering the hyper parameter. In the experimental section the proposed algorithms are applied on an artificial and a real life dataset using Fuzzy Cohen's Kappa to compare the results.

Theoretical considerations lead to the fuzzy variants of Neural Gas and Self Organizing Maps, proposing two semi-supervised classifiers. For the Fuzzy Labeled Neural Gas three different ways are derived to incorporate the additional class information. Further, is is shown, that relevance learning can easily be included to improve the classification accuracy.

Classification in machine learning is an algorithmic procedure, where given input data is assigned to one (crisp) or more (fuzzy) known categories, also referred to as classes. It is a supervised learning scheme since the training data is labeled, i.ẽ. the respective classes of the training data are known. This information is usually

obtained with the help of some experts, who manually label the data. Well known classifiers are Support Vector Machines (SVM), Decision Trees, and Multi-Layer Perceptrons (MLP). Here the focus is laid on prototype based classification procedures, i. e. each class is represented by one or more prototypes which are located within the same metric space as the data samples.

Nearest Prototype Classification (NPC) (Kohonen 1995) is a method where a set of class dependent prototypes is positioned to optimize the classification of the data points according to their distances. After training, unlabeled data samples are uniquely (crisp) assigned to the class of the closest protoype. A well known and widely used learning scheme using Nearest Prototype Classification is the Learning Vector Quantization (LVQ) as introduced by KOHONEN (Kohonen 1986). The original version is the basis for a whole family of supervised learning algorithms like LVQ 2.1 (Kohonen 1986), GLVQ (Sato and Yamada 1996) and GRLVQ (Hammer and Villmann 2002) to name just a few. There are also further modified methods resulting in soft (fuzzy) classifications like SLVQ and RSLVQ by SEO & OBERMAYER (Seo and Obermayer 2003).

SEO ET AL. also developed a soft version for the Nearest Prototype Classification called SNPC which uses the Gaussian mixture approach to model soft assignments of the data points to their representing prototypes (Seo et al. 2003). This method utilizes stochastic gradient descent on a cost function incorporating the probability density of the data points and can be interpreted as an annealed version of LVQ.

Yet all of the above mentioned algorithms have in common that they rely on the major assumption that the training data and prototypes are uniquely associated with one specific class. VILLMANN ET AL. established a new learning scheme called FSNPC which is just as SNPC based on the Gaussian mixture model but utilizes fuzzy prototype vectors (Villmann, Schleif and Hammer 2006). While SNPC prototypes realize only a crisp classification because of their unique class dependence, FSNPC prototypes are capable of representing overlapping classes which cannot be described appropriately by crisp prototypes.

Yet still, the training data samples inherit crisp class assignments, i. e. each data point for the training is known to belong to exactly one class. But in practice this might not be a realistic assumption. For example, in medicine a patients disease might not be uniquely classifiable to only one diagnosis and the medical doctor implicitly makes probability assumptions about the true kind of illness. Or the identification of cancer or biological specimen based on tissue samples allows only a diffuse classification. Sometimes the training data itself can only be obtained by insecure methods due to technical reasons, subjective evaluation procedures, additive noise or missing information. Therefore, learning algorithms for prototype based classifiers handling uncertain class assignments of the training data are required.

Note, that contrary to fuzzy clustering, where only the final assignments of the data samples to the prototypes are fuzzy, fuzzy classification implies, that within the dataset used to perform the training an amount of uncertainty is contained.

The current chapter is divided into two parts: supervised learning and semi-supervised learning. In the first part, two classifiers based on fuzzy labeled training data are presented. Section 4.1.1 introduces an extension of the FSNPC incorporating uncertainty of the training set. The learning scheme is based on a Gaussian mixture model and fuzzy prototypes. For the robust gradient based RSLVQ, which maximizes the likelihood ratio, a vectorial adaptation scheme for fuzzy labeled training data is proposed in section 4.1.2.

Self-Organizing Maps (SOM) (Kohonen 1990) and Neural Gas (NG) (Martinetz et al. 1993), described in detail in sections 2.1.3 and 2.1.4, are also prototype based vector quantizers, but since they work with unlabeled training data they can be used for clustering only. Yet for both of them exist variants which incorporate class information in the training process and this way realize a semi-supervised learning. These approaches range from Supervised NG (SNG) (Hammer et al. 2005) over simple post labeling for SOM (de Wouwer et al. 1996) to the well-known counterpropagation network (Hecht-Nielsen 1987, Hecht-Nielsen 1988). However, all of these methods have in common, that the locations of the prototypes in the data space remain unchanged by the subsequent determination of the prototype labels. In section 4.2.1 an extension of the SOM is presented, where the positions of the prototypes are explicitly influenced by the classification task. Further, the labels of the training data are allowed to be fuzzy. And finally section 4.2.2 provides a modification of the NG. By incorporating fuzzy class information in the learning process fuzzy classification can be performed.

A further aspect is relevance learning. As mentioned before, commonly the squared Euclidean distance is chosen to obtain the dissimilarity of data points and prototypes. Yet, assuming a distance with an additional parameter emphasizing or weakening particular input dimensions, it might be useful to adapt this relevance parameter as well. Thereby, without almost no further expenses, an increased flexibility of the model in combination with an improved interpretability of the results will be gained. By learning the relevance parameter besides the prototypes and their fuzzy labels further improvements with regard to flexibility and classification ability can be achieved. In sections 4.2.1 and 4.2.2 this idea is picked up and the theoretical aspects of relevance learning are considered. The learning rule is obtained by an optimization of the respective cost functions with respect to the relevance parameter.

4.1 Supervised learning

In this section a variant of a Nearest Prototypes Classifier as well as an extension to RSLVQ are proposed. Since both of them can be applied to fuzzy labeled data sets, they are called F-FSNPC and FRSLVQ, where the *F* stands for *fuzzy*.

The abilities of the resulting algorithms are demonstrated on a two-dimensional artificial data set consisting of two overlapping Gaussian distributions and on a real world problem classifying barley grain tissue.

4.1.1 Classifying fuzzy labeled data with FSNPC

The FSNPC from the family of nearest neighbor classifiers was proposed by VILLMANN ET AL. (Villmann, Schleif and Hammer 2006) and is summarized in section 2.2.6. The main characeristic is its ability to generate fuzzy labeled prototypes based on crisp class assignments of the training data. Given the fuzzy prototypes it is possible to obtain *probabilistic* class assignments for unknown data points.

The here described extension F-FSNPC was first proposed in (Geweniger and Villmann 2010) and also generates fuzzy prototypes, but contrary to FSNPC it takes fuzzy class labels of the training data into account, i. e. a certain level of uncertainty is implied within the data set itself. An appropriate learning scheme based on a Gaussian mixture model has been developed to update the prototypes.

Cost function

To adapt the cost function of the F-FSNPC to fuzzy labeled training data, the crisp class assignments c_i of the data points $\boldsymbol{v}_i$ were remodeled to a N_C-dimensional probabilistic vector $\boldsymbol{c}_i$ with $\sum_{k=1}^{N_C} c_i^k = 1$ and $c_i^k \geq 0$ where N_C is the number of classes. Hence, we consider a training data set $\mathcal{S} = \{(\boldsymbol{v}_i, \boldsymbol{c}_i)\}_{i=1}^{N_V}$ with fuzzy class assignments $\boldsymbol{c}_i$. The function of the local costs eq. (2.69) now takes the proportionate class assignments $\boldsymbol{c}_i$ of the training data points $\boldsymbol{v}_i$ into account and is therefore changed to

$$lc_i = \sum_{j=1}^{N_P} P(j|\boldsymbol{v}_i)(\boldsymbol{c}_i - \boldsymbol{y}_j)^2 \tag{4.1}$$

where $(\boldsymbol{c}_i - \boldsymbol{y}_j)^2$ is used as an abbreviation for $(\boldsymbol{c}_i - \boldsymbol{y}_j)^T(\boldsymbol{c}_i - \boldsymbol{y}_j)$. Note, that for the local cost yields $lc_i \leq 1$, because $\boldsymbol{c}_i$ and $\boldsymbol{y}_j$ are both probabilistic class assignments. The cost function (2.69), as the sum over the local costs of all the data points, will

not change, except that now the local costs according to eq. (4.1) are considered

$$\begin{aligned} E_{F-FSNPC} &= \frac{1}{N_V}\sum_{i=1}^{N_V} lc_i \\ &= \frac{1}{N_V}\sum_{i=1}^{N_V}\sum_{j=1}^{N_P} P(j|\boldsymbol{v}_i)(\boldsymbol{c}_i - \boldsymbol{y}_j)^2 \end{aligned} \tag{4.2}$$

Update rules

As usual, the F-FSNPC prototype update rules are obtained as stochastic gradient descent of the cost function (4.2) with respect to the prototypes $\boldsymbol{w}_j$. It can be observed, that the cost function differs from that of FSNPC (2.61) only by the now fuzzy data assignments $\boldsymbol{c}_i$ instead of the crisp assignments. However, as in FSNCP and SNPC these data class assignments do not depend on the prototypes $\boldsymbol{w}_j$. Therefore, the update rules are not affected by this substitution. The update rules for F-FSNPC are structurally the same as for FSNPC (2.68), but now the crisp data assignments are replaced by the fuzzy data assignments which leads to

$$\Delta\boldsymbol{w}_j = -\frac{\varepsilon}{2\sigma^2} P(j|\boldsymbol{v}_i)\left((\boldsymbol{c}_i - \boldsymbol{y}_j)^2 - lc_i\right)\frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j}. \tag{4.3}$$

The update rule for the fuzzy prototype labels $\boldsymbol{y}_j$ is obtained in the same manner. Considering the stochastic gradient descent of the cost function (4.2) with respect to $\boldsymbol{y}_j$ yields

$$\Delta\boldsymbol{y}_j = -2P(j|\boldsymbol{v}_i)(\boldsymbol{c}_i - \boldsymbol{y}_j). \tag{4.4}$$

In complete analogy to FSNPC in section 2.2.6, a window rule for the active region for the prototype update can be derived. By setting $T = P(j|\boldsymbol{v})((\boldsymbol{c}_i - \boldsymbol{y}_j)^2 - lc_i)$ and using the Gaussian form (2.65) for $P(j|\boldsymbol{v})$ the term T can be rewritten as $T = T_0 \cdot \Pi(\boldsymbol{v}_i, \boldsymbol{c}_i, \boldsymbol{w}_j, \boldsymbol{y}_j)$ with

$$T_0 = ((\boldsymbol{c}_i - \boldsymbol{y}_j)^T(\boldsymbol{c}_i - \boldsymbol{y}_j))^2 - lc_i^2 \tag{4.5}$$

and

$$\Pi(\boldsymbol{v}_i, \boldsymbol{c}_i, \boldsymbol{w}_j, \boldsymbol{y}_j) = \frac{e^{\left(-d(\boldsymbol{v}_i, \boldsymbol{w}_j)/(2\sigma^2)\right)}}{\sum_k \left((\boldsymbol{c}_i - \boldsymbol{y}_j)^2 + (\boldsymbol{c}_i - \boldsymbol{y}_k)^2\right) e^{(-d(\boldsymbol{v}_i, \boldsymbol{w}_k)/(2\sigma^2))}}. \tag{4.6}$$

For the complete derivation of (4.5) and (4.6) refer to appendix 4.A. Obviously, $lc_i^2 \leq 1$ because $lc_i \leq 1$. Further, because all components of $\boldsymbol{c}_i$ and $\boldsymbol{y}_j$ are less or equal to one and greater or equal to zero, $(\boldsymbol{c}_i - \boldsymbol{y}_j)^T(\boldsymbol{c}_i - \boldsymbol{y}_j) \leq K$, where K is a data

dependend constant. Hence, we have $-K^2 \leq T_0 \leq K^2 + 1$. The absolute value of T_0 has to be significantly different from zero to have a valuable contribution to the update. Therefore $|T_0| \gg 0$ defines a window rule as it is known from SNPC (Seo et al. 2003) and LVQ (Kohonen 1986), see also sections 2.2.5 and 2.2.1.

4.1.2 Fuzzy Robust Soft LVQ

Robust Soft Learning Vector Quantization (RSLVQ) as proposed by SEO & OBERMAYER (Seo and Obermayer 2003) is a robust classification algorithm based on a likelihood function incorporating probability densities, see section 2.2.4 for details. Yet this algorithm is only applicable for crisp labeled training data. An extension of this approach based on a vectorial adaption scheme for handling fuzzy labeled training data is presented in this chapter. The focus is laid on the adaption of the protoypes and the influence of fuzzy data assignments on the learning behaviour in general. Further aspects, in particular hyper parameter learning, have been presented and discussed by SCHNEIDER (Schneider 2010).

After the theoretical analysis this section conludes with two experiments based on an artificial data set and a real world problem.[1]

Cost function

The assumption of fuzzy labeled data points requires an adaption of the original RSLVQ algorithm. The originally crisp class label c_i for training data point $\boldsymbol{v}_i$ becomes a N_C-dimensional vector $\boldsymbol{c}_i$ of assignment probabilities with $\sum_{k=1}^{N_C} c_i^k = 1$ and $c_i^k \geq 0$. As for the RSLVQ, each prototype $\boldsymbol{w}_j$ describes exactly one class. But to be conform with the notation for the data points, the class membership of the prototypes is now also expressed in vector notation yielding $\boldsymbol{y}_j$ with $\sum_{k=1}^{N_C} y_j^k = 1$ and $y_j^k \geq 0$. The classification of untrained data is still based on the winner takes all scheme (2.45). Taking the fuzzy class assignments of the data points into account, the particular probability density $p(\boldsymbol{v}_i, c_i|W)$ with crisp data labels c_i specified in equation (2.55) changes to

$$p(\boldsymbol{v}_i, \boldsymbol{c}_i|W) = \sum_{k=1}^{N_C} c_i^k \sum_{j=1}^{N_P} y_j^k \cdot p(\boldsymbol{v}_i|j)P(j) \tag{4.7}$$

where $p(\boldsymbol{v}_i, \boldsymbol{c}_i|W)$ now is the particular probability density that data point $\boldsymbol{v}_i$ is generated by the mixture components referred to by $\boldsymbol{c}_i$. Thereby, due to the factor c_i^k

[1] Special thanks to Petra Schneider for conducting the experiments.

only a fraction of the sum of the respective probability densities is taken into account. The factor y_j^k ensures that only those prototypes are accounted for, which actually are representatives for the respective class.

The total probability density $p(\boldsymbol{v}_i|W)$ (2.56)

$$p(\boldsymbol{v}_i|W) = \sum_j p(\boldsymbol{v}_i|j)P(j)$$

is the probability that data point $\boldsymbol{v}_i$ is generated by *any* prototype. It is the sum over all prototypes independent of matching class assignments and, therefore, does not change.

The cost function of the Fuzzy RSLVQ (FRSLVQ) can now be defined as

$$E_{FRSLVQ} = \sum_{i=1}^{N_V} \log\left(\frac{p(\boldsymbol{v}_i, \boldsymbol{c}_i|W)}{p(\boldsymbol{v}_i|W)}\right). \tag{4.8}$$

Derivation of learning rules

In order to optimize the classification, the cost function (4.8) has to be maximized, which can be done by a stochastic gradient ascent with respect to the parameter to update (Robbins and Monro 1951).

Considering an universal parameter Θ with $\Theta \neq \boldsymbol{v}_i$ a general update rule (4.9) can be derived:

$$\frac{\partial \log \frac{p(\boldsymbol{v}_i, \boldsymbol{c}_i|W)}{p(\boldsymbol{v}_i|W)}}{\partial \Theta_j} = (P_{\boldsymbol{c}_i}(j|\boldsymbol{v}_i) - P(j|\boldsymbol{v}_i))\left(\frac{1}{K(j)}\frac{\partial K(j)}{\partial \Theta_j} + \frac{\partial f(\boldsymbol{v}_i, \boldsymbol{w}_j, \sigma_j^2))}{\partial \Theta_j}\right) \tag{4.9}$$

A detailed description of the derivation process can be found in appendix 4.B. To obtain the update rules for specific, cost function relevant parameters, Θ_j has to be substituted. The terms $P_{\boldsymbol{c}_i}(j|\boldsymbol{v}_i)$ and $P(j|\boldsymbol{v}_i)$ in (4.9), which are the assignment probabilities, yield:

$$P_{\boldsymbol{c}_i}(j|\boldsymbol{v}_i) = \frac{\boldsymbol{c}_i^{c(\boldsymbol{y}_j)} P(j)K(j)e^{f(\boldsymbol{v}_i, \boldsymbol{w}_j, \sigma_j^2, \lambda_j)}}{p(\boldsymbol{v}_i, \boldsymbol{c}_i|W)} \tag{4.10}$$

$$P(j|\boldsymbol{v}_i) = \frac{P(j)K(j)e^{f(\boldsymbol{v}_i, \boldsymbol{w}_j, \sigma_j^2, \lambda_j)}}{p(\boldsymbol{v}_i|W)} \tag{4.11}$$

$P_{\boldsymbol{c}_i}(j|\boldsymbol{v}_i)$ is the assignment probability of $\boldsymbol{v}_i$ to component j within class $c(\boldsymbol{y}_j)$, where $c(\boldsymbol{y}_j)$ is a function yielding the index k of this vector component, for which

$y_j^k = 1$ with $k \in \{1, \ldots, N_C\}$. I. e. $c(\boldsymbol{y}_j)$ is the class assignment of prototype $\boldsymbol{w}_j$ and $\boldsymbol{c}_i^{c(\boldsymbol{y}_j)}$ the fuzzy class membership of data point $\boldsymbol{v}_i$ to class class $c(\boldsymbol{y}_j)$. $P(j|\boldsymbol{v}_i)$ is the assignment probability of $\boldsymbol{v}_i$ to component j independent of the class membership.

Assuming the special case of a Gaussian mixture model with $P(j) = 1/N_P \forall j$, the similarity function is set to $f(\boldsymbol{v}_i, \boldsymbol{w}_j, \sigma_j^2) = \frac{d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{2\sigma_j^2}$. Thereby, $d(\boldsymbol{v}_i, \boldsymbol{w}_j)$ is the distance between data point $\boldsymbol{v}_i$ and prototype $\boldsymbol{w}_j$, and $K(j)$ a normalization constant which can be set to $K(j) = (2\pi\sigma_j^2)^{(-N/2)}$.

The original RSLVQ algorithm uses the squared Euclidean distance as dissimilarity measure. In the following the update rules for the prototypes $\boldsymbol{w}_j$ and a hyper parameter σ_j^2 employing a general distance are derived. Afterwards the update rules based on spezific distance measures are given.

Updating the prototypes $\boldsymbol{w}$

To obtain the update rule for the prototypes the general parameter Θ_j in (4.9) has to be replaced by the prototype $\boldsymbol{w}_j$:

$$\frac{\partial \log \frac{p(\boldsymbol{v}_i, \boldsymbol{c}_i|W)}{p(\boldsymbol{v}_i|W)}}{\partial \boldsymbol{w}_j} = (P_{\boldsymbol{c}_i}(j|\boldsymbol{v}_i) - P(j|\boldsymbol{v}_i)) \left(\frac{1}{K(j)} \frac{\partial K(j)}{\partial \boldsymbol{w}_j} - \frac{\partial f(\boldsymbol{v}_i, \boldsymbol{w}_j, \sigma_j^2)}{\partial \boldsymbol{w}_j} \right). \quad (4.12)$$

Since $K(j)$ is independent of $\boldsymbol{w}_j$, the partial derivate $\partial K(j)/\partial \boldsymbol{w}_j$ evaluates to zero. The partial derivative of the similarity function $f(\boldsymbol{v}_i, \boldsymbol{w}_j, \sigma_j^2)$ yields for the Gaussian mixture model

$$\frac{\partial f(\boldsymbol{v}_i, \boldsymbol{w}_j, \sigma_j^2)}{\partial \boldsymbol{w}_j} = \frac{1}{2\sigma_j^2} \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j} \quad (4.13)$$

and therefore

$$\frac{\partial \log \frac{p(\boldsymbol{v}_i, \boldsymbol{c}_i|W)}{p(\boldsymbol{v}_i|W)}}{\partial \boldsymbol{w}_j} = (P_{\boldsymbol{c}_i}(j|\boldsymbol{v}_i) - P(j|\boldsymbol{v}_i)) \left(\frac{1}{2\sigma_j^2} \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j} \right) \quad (4.14)$$

is obtained.

The original crisp RSLVQ algorithm is based on the squared Euclidean distance. For the here described fuzzy labeled variant the respective update rule yields

$$\Delta \boldsymbol{w}_j^{Euclid} = -\frac{\varepsilon_1}{\sigma^2} (P_{\boldsymbol{c}_i}(j|\boldsymbol{v}_i) - P(j|\boldsymbol{v}_i)) (\boldsymbol{v}_i - \boldsymbol{w}_j) \quad (4.15)$$

for each prototype $\boldsymbol{w}_j$ and with learning rate $\varepsilon_1 > 0$. Yet, other metric distance measures or divergences might be used as well. For example, the update rules based

on the Generalized Kullback–Leibler divergence (2.84) and the Generalized Rényi divergence (2.81) are

$$\Delta \boldsymbol{w}_j^{GKL} = \frac{\varepsilon_1}{\sigma^2} \left(P_{\boldsymbol{c}_i}(j|\boldsymbol{v}_i) - P(j|\boldsymbol{v}_i)\right) \left(1 - \frac{\boldsymbol{v}_i}{\boldsymbol{w}_j}\right) \tag{4.16}$$

$$\Delta \boldsymbol{w}_j^{GR,\alpha} = \frac{\varepsilon_1}{\sigma^2} \left(P_{\boldsymbol{c}_i}(j|\boldsymbol{v}_i) - P(j|\boldsymbol{v}_i)\right) \left(\frac{\boldsymbol{v}_i^\alpha \boldsymbol{w}_j^\alpha - 1}{1 + \sum_{k=1}^d \left[\boldsymbol{v}_i^\alpha \boldsymbol{w}_j^{1-\alpha} - \alpha \boldsymbol{v}_i - (1-\alpha)\boldsymbol{w}_j\right]}\right). \tag{4.17}$$

Updating the hyper parameter $\boldsymbol{\sigma^2}$

The hyper parameter update rule can be deduced by replacing Θ_j in (4.9) with σ_j^2:

$$\frac{\partial \log \frac{p(\boldsymbol{v}_i, \boldsymbol{c}_i|W)}{p(\boldsymbol{v}_i|W)}}{\partial \sigma_j^2} = \left(P_{\boldsymbol{c}_i}(j|\boldsymbol{v}_i) - P(j|\boldsymbol{v}_i)\right) \left(\frac{1}{K(j)} \frac{\partial K(j)}{\partial \sigma_j^2} - \frac{\partial f(\boldsymbol{v}_i, \boldsymbol{w}_j, \sigma_j^2)}{\partial \sigma_j^2}\right). \tag{4.18}$$

The partial derivatives of $K(j)$ and $f(\boldsymbol{v}_i, \boldsymbol{w}_j, \sigma_j^2)$ with respect to the hyper parameter σ_j^2 are

$$\frac{\partial K(j)}{\partial \sigma_j^2} = -\frac{N}{2} \frac{1}{(2\pi\sigma_j^2)^{N/2}\, \sigma_j^2} \tag{4.19}$$

$$\frac{\partial f(\boldsymbol{v}_i, \boldsymbol{w}_j, \sigma_j^2)}{\partial \sigma_j^2} = \frac{d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{2\,\sigma_j^4} \tag{4.20}$$

and in combination with eq. (4.18) learning rule

$$\Delta \sigma_j^2 = \varepsilon_2 \left(P_{\boldsymbol{c}_i}(j|\boldsymbol{v}_i) - P(j|\boldsymbol{v}_i)\right) \left(\frac{d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\sigma_j^4} - \frac{N}{\sigma_j^2}\right) \tag{4.21}$$

is obtained. The parameter ε_2 is the learning rate for the hyper parameter.

For the more general case of the global parameter $\sigma_j^2 = \sigma^2$ being identical for all components j, the hyper parameter can be updated by the summation of the probability assignments

$$\Delta \sigma^2 = \varepsilon_2 \sum_{j=1}^m \left(P_{\boldsymbol{c}_i}(j|\boldsymbol{v}_i) - P(j|\boldsymbol{v}_i)\right) \frac{d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\sigma_j^4}. \tag{4.22}$$

Further details concerning the hyper parameter and various aspects thereof can be found in (Schneider 2010).

4.1.3 Artificial example – overlapping Gaussian distributions

The data set consists of two spherical Gaussian clusters of equal variance in a two-dimensional space. Each cluster consists of 1000 samples with their mean values set to $\mu_1 = [-1, 0]$ and $\mu_2 = [1, 0]$. Five different settings for the variance $\varphi^2 \in \{0.1, 0.3, 0.5, 0.7, 1.0\}$ are chosen. The class memberships c_i^1 and c_i^2 of sample $\boldsymbol{v}_i$ are defined depending on the first component v_i^1. For the region between the means $-1 \leq v_i^1 \leq 1$ a linear relationship was chosen, whereas data points $v_i^1 < -1$ and $v_i^1 > 1$ outside of the overlap where assigned crisp, see also Fig. 4.1 (left). The value v_i^2 is irrelevant for the labeling.

F-FSNPC training For the analysis of F-FSNPC several runs with two prototypes and varying variances have been performed. The initial class assignments of the prototypes were set to 50% for each class. At each training step the prototypes and their fuzzy class labels were updated. Note, that the prototypes are allowed to switch their affiliation to the classes.

It can be observed, that contrary to FSNPC with crisp labeled training data, the prototypes now behave differently. While during FSNPC training the prototypes get positioned near the decision boundary, F-FSNPC tends to move them into those regions of the data space, where there is a higher degree of classification agreement within the data set, see Fig. 4.1 (right). The less fuzzy the labels of the data points, the more attractive is this region to the prototypes.

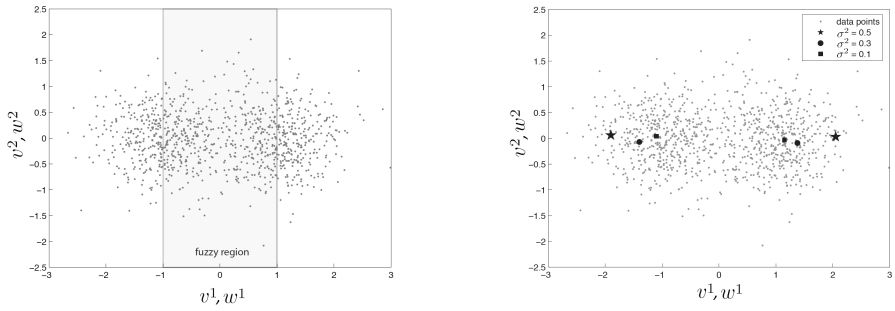

Figure 4.1: *Artificial data. Left: Two-dimensional Gaussian distributions with highlighted fuzzy region between the centers. Right: Mean final prototype locations after F-FSNPC training with varying softness on data sets with* $\varphi^2 = 0.3$

FRSLVQ training Considering FRSLVQ, the experiments are more complex, since a further hyper parameter σ^2 is taken into account. Therefore, the experiment is split into three parts:

- learn one prototype per class with constant hyper parameter σ^2
- optimize the hyper parameter σ^2 with the prototypes being fixed in the cluster centers
- learn hyper parameter σ^2 and prototypes simultaneously

The findings are compared to identical experiments with RSLVQ. For the analysis, the learning parameters are set to $\varepsilon_1 = 1 \cdot 10^{-3}$ and $\varepsilon_2 = 5 \cdot 10^{-5} \cdot \sigma^2(0)$ and the fixed and initial values of the hyper parameter to $\sigma^2(0) \in \{0.05, 0.15, 0.3\}$. The prototypes are initialized close to the cluster means and training is continued for 500 epochs. Each experiment is performed on ten independent data sets.

Learning the prototypes with constant σ^2 During FRSLVQ training with constant σ^2, the prototypes move away from each other; they move along the first axis away from the cluster centers. The final distance $\|\boldsymbol{w}_1 - \boldsymbol{w}_2\|$ depends on the value of the hyper parameter and the cluster's variance. Namely, the distance increases with increasing softness and increasing variance φ^2; though, the influence of φ^2 is comparably weak. These observations are depicted in Fig. 4.2. On the contrary, the opposite effect is observed during RSLVQ training, i. e., the prototypes move in the direction of the decision boundary; the distance between $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ decreases during training. The prototypes saturate closer to the decision boundary the larger φ^2 and the smaller σ^2, see Fig. 4.3.

Optimizing the hyper parameter with fixed prototypes The results of hyper parameter learning with fixed prototypes are visualized in Fig. 4.4. FRSLVQ converges to values very close to zero after only a small number of training epochs. On the other hand, RSLVQ approaches the clusters' variance. These observations hold for both algorithms independent of the initialization $\sigma^2(0)$.

Simultaneous learning of prototypes and hyper parameter Concerning FRSLVQ, the simultaneous training of prototypes and hyper parameter initially shows the same behavior as described above: the prototypes move away from each other and the hyper parameter quickly converges to zero. However, the prototypes' movement is stopped, when σ^2 reaches very small values. Hence, the distance between the prototypes does not increase that extensively as observed in the experiments

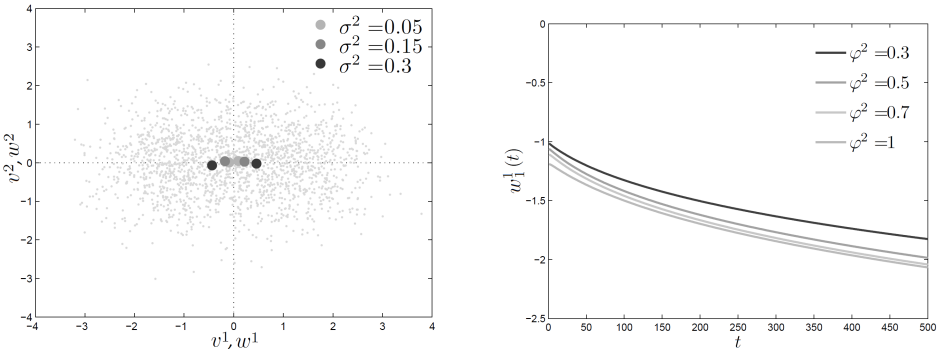

Figure 4.2: *Artificial data. Left: Mean final prototype locations after FRSLVQ training with varying softness on data sets with $\varphi^2 = 0.3$. Right: Trajectories of first component of class one prototype during FRSLVQ training on datasets with different cluster variance and equal hyper parameter $\sigma^2 = 0.15$.*

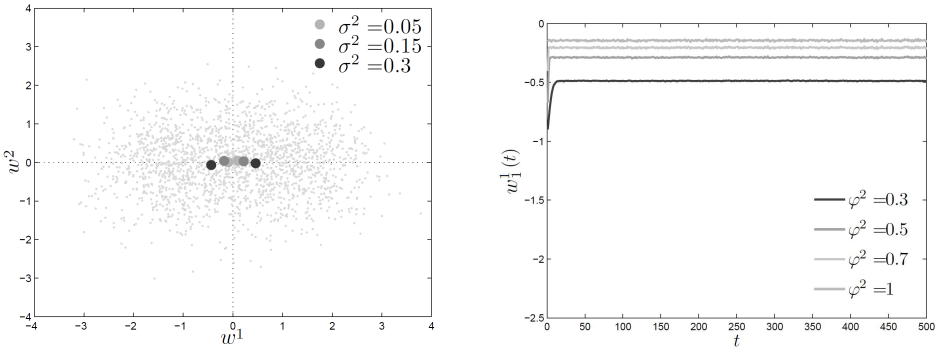

Figure 4.3: *Artificial data. Left: Mean final prototype locations after RSLVQ training with varying σ^2 on data sets with $\varphi^2 = 0.3$. Right: Trajectories of first component of class one prototype during RSLVQ training on datasets with different cluster variances and equal hyper parameter $\sigma^2 = 0.15$.*

with constant σ^2. The hyper parameter training in RSLVQ weakens the movement of prototypes towards to decision boundary; σ^2 reaches smaller values compared to the experiments with fixed prototypes.

Comparing F-FSNPC with FRSLVQ To compare the cluster solutions obtained by F-FSNPC and FRSLVQ, Fuzzy Cohen's Kappa $\widehat{\kappa}_F$ has been considered (Cohen 1960). This index, described also in section 2.4.3, measures the agreement between two

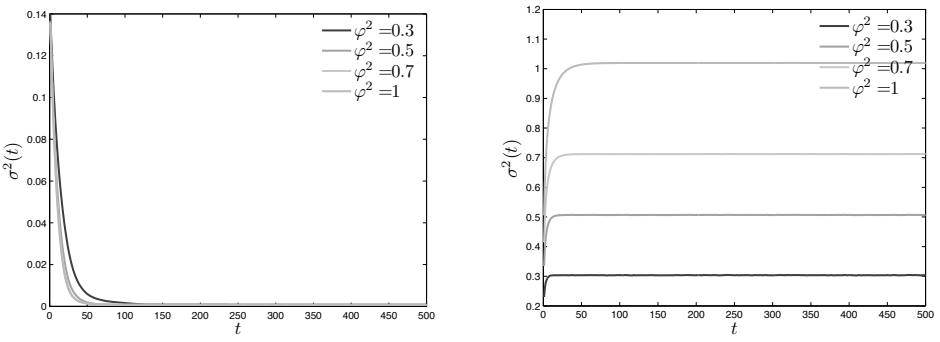

Figure 4.4: *Artificial data. Mean evolution of the hyper parameter during training on datasets with different cluster variances φ^2 and constant prototypes fixed in the cluster centers. The hyper parameter was always initialized with $\sigma^2(0) = 0.15$. The plots are representative for all tested $\sigma^2(0)$. Left: FRSLVQ-Training. Right: RSLVQ-Training.*

classifiers. According to SACHS (Sachs 1992) the value of the index can be interpreted in terms of *perfect agreement*, *substantial agreement*, and so on up to *agreement by chance*. Comparing the F-FSNPC cluster solutions with those obtained by FRSLVQ with equivalent settings, frequently yields *substantial* agreements. The Kappa index regarding the comparison of the clustering agreements with the original data, *perfect* to *substantial* agreements are obtained. As expected, the index for the training data was frequently higher than for the test data.

4.1.4 Real world example – barlay grain tissue sections

The real life data set is based on serial transverse sections of barley grains at different developmental stages, see Fig. 4.5, and was used before in (Brüß et al. 2006) and (Villmann et al. 2007). The classification task consists in the identification of 11 different tissue types like nuclear epidermis, transfer cell, and chlorophyll layer. The classification of these samples was done manually and especially for border tissue as depicted in Fig. 4.5 (right) there was no distinct type identification possible. For this reason, fuzzy class assignments are provided beside the crisp labeling. The samples are described by means of 144 features and 4418 data points are available, about half of them have fuzzy labels. The data set is randomly split into 3800 samples for training and 618 for testing purposes.

Training For means of comparison the training was conducted for F-FSNPC and FRSLVQ along with the crisp variants FSNPC and FRSLVQ. In all experiments, one

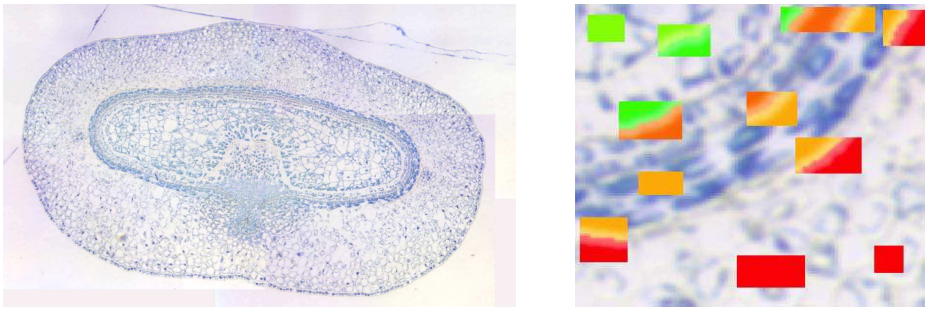

Figure 4.5: *Grain data set. Left: Example of a barley grain tissue sample. Right: Section with uncertain class assigments.*

prototype per class and for the LVQ variant a global hyper parameter are adapted to the data. The learning rates are set to $\varepsilon_1 = 0.01$ and $\varepsilon_2 = 5 \cdot 10^{-4} \cdot \sigma^2(0)$, where the initial value of the hyper parameter is chosen as $\sigma^2(0) \in \{0.5,\ 1.0,\ 2.0\}$. The training is run for 200 epochs and to verify the results, the experiments are repeated on five independent constellations of training set and test set.

The algorithms identify nearly the same prototypes as exemplarily depicted in Fig. 4.6 for FRSLVQ and RSLVQ and the prototypes representing class 5. Also, the learning process of the hyper parameter is almost identical comparing FRSLVQ and RSLVQ. The observed differences are not as drastic as in the previous experiments. The evolution of σ^2 in the course of FRSLVQ and RSLVQ training for the different initializations $\sigma^2(0)$ is depicted in Fig. 4.7. The curves yielded by the alternative algorithms resemble to large extent, that σ^2 increases slightly at the beginning of training, but starts degrading after approximately 10 sweeps through the training set. Finally the hyper parameter always converges to the same value, independent of $\sigma^2(0)$. Yet it has to be noted, that the final value $\sigma^2(t)$ is slightly smaller after FRSLVQ training; we observe $\sigma^2_{FRSLVQ}(t) \approx 0.9$ and $\sigma^2_{RSLVQ}(t) \approx 1.3$. Obviously, the uncertainty in the class memberships induces smaller optimal values σ^2.

Adding noise To verify this last assumption concerning the hyper parameter, the fuzziness of the class labels is artificially increased by adding uniform noise of different variance 0.1, 0.3, and 0.5 to the label vectors, followed by a normalization step to guarantee $\sum_{k=1}^{d} c^k = 1$. The FRSLVQ training process is repeated with the same learning parameters as before. As depicted in Fig. 4.8, $\sigma^2(t)$ approaches smaller values with increasing noise level.

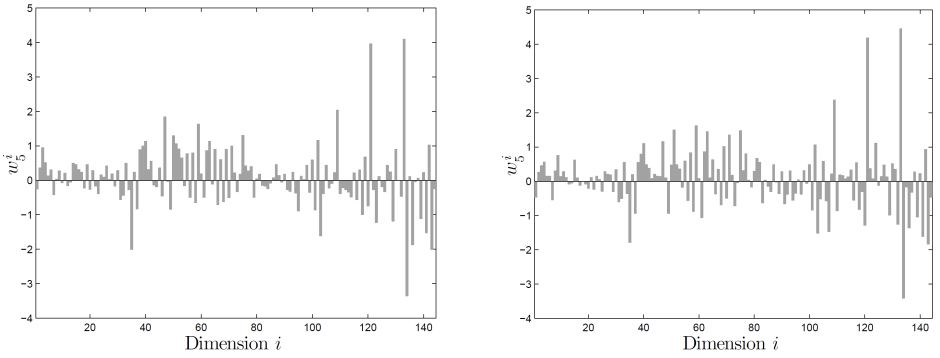

Figure 4.6: *Grain data set. Visualization of the class 5 prototypes obtained by FRSLVQ training (left) and RSLVQ training (right) in one training run.*

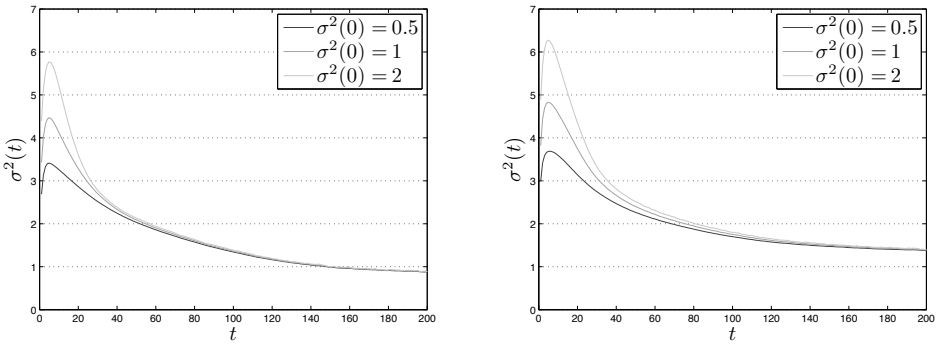

Figure 4.7: *Grain data set. Mean evolution of the hyper parameter during training with different initial settings $\sigma^2(0)$. Left: FRSLVQ training. Right: RSLVQ training.*

Evaluation In order to evaluate the classification accuracy, Fuzzy Cohen's Kappa $\widehat{\kappa}_F$ as introduced in (Dou et al. 2007) and described in section 2.4.3 is computed. The evolution of the coefficient $\widehat{\kappa}_F$ during FRSLVQ training on the original data set is depicted in Fig. 4.9. It reaches $\widehat{\kappa}_{FRSLVQ}^{train} \approx 0.83$ which corresponds to perfect agreement; the final value on the test data $\widehat{\kappa}_{FRSLVQ}^{test} \approx 0.78$ implies substantial agreement (Fig. 4.9, left). The additional noise in the class labeling clearly reduces the algorithm's performance. With the lowest noise level we applied in our testings, both values $\widehat{\kappa}_{FRSLVQ}^{train}$ and $\widehat{\kappa}_{FRSLVQ}^{test}$ degrade to only moderate agreement (Fig. 4.9, right). Comparing F-FSNPC with FRSLVQ yields $\widehat{\kappa}_{F-FSNPC}^{FRSLVQ} \approx 0.72$.

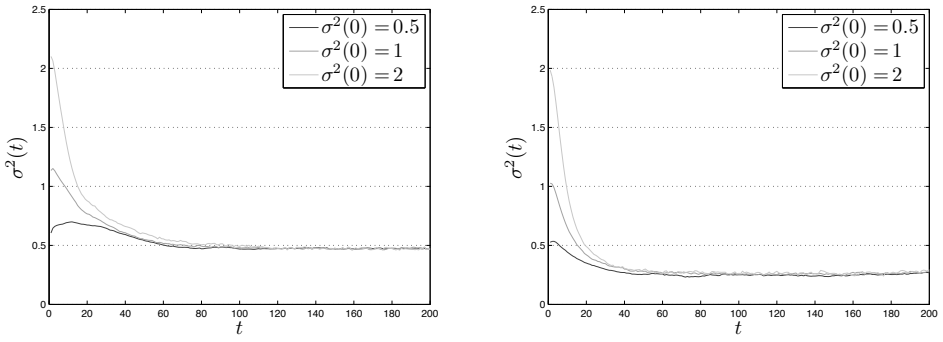

Figure 4.8: *Grain data set. Mean evolution of the hyper parameter during FRSLVQ training with different initial settings $\sigma^2(0)$. The fuzziness of the original data set was increased by adding random noise to the class labels. Left: Uniform noise with variance 0.1 added. Right: Uniform noise with variance 0.5 added.*

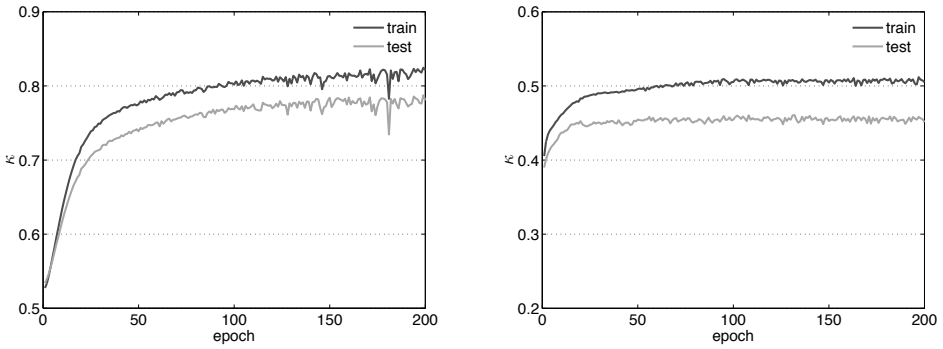

Figure 4.9: *Grain data set. Mean evolution of coefficient κ_F on training and test data during FRSLVQ training with adaptive softness and $\sigma^2(0) = 1$. The plot is representative for all initializations $\sigma^2(0)$. Left: Training on original data set. Right: Training on data set with increased fuzziness.*

4.1.5 Conclusion

FSNPC proposed in (Villmann, Schleif and Hammer 2006) is a classifier for crips data yet working with fuzzy labeled prototypes. Now it has been modified to handle fuzzy labeled training data as well. The rules for the prototype update as well as the update of their fuzzy class assignments were derived by a stochastic gradient descent on the cost function, which incorporates the fuzzy data assignments.

The known RSLVQ algorithm (Seo and Obermayer 2003) has been extended to

work with uncertain class labels. This new variant, where the crisp class assignment of each prototype is replaced by a probabilistic vector reflecting the relative class assignments, is called Fuzzy RSLVQ (FRSLVQ). Update rules for the prototypes and the hyper parameter were derived and in extensive experiments the behavior of the learning process was analyzed.

Comparing the results of FSNPC and RSLVQ with those obtained by the new F-FSNPC and FRSLVQ yields several differences:

- Using FSNPC or RSLVQ the prototypes converge towards the decision boundary, yet for the two new algorithms F-FSNPC and FRSLVQ the prototypes tend to move away from each other into a region of higher classification accuracy. This is due to the fact that the contribution of the data points to the prototype update for a specific class depends on their strength for describing this specific class. The higher the classification accuracy of the data points the higher their attraction for the prototypes.

- For FRSLVQ the initial choice of the hyper parameter σ^2 has no influence on its final value. For RSLVQ the hyper parameter approaches the cluster's variance, whereas for FRSLVQ this parameter converges to rather small or even close to zero values during the learning process depending on the degree of uncertainty within the data set. The prototype update stops when the hyper parameter reaches very small values.

- Using real life data, F-FSNPC and FRSLVQ show the same behavior as with the artificial dataset with the minor difference, that the results are not as strongly pronounced. The FRSLVQ hyper parameter converges to small values but does not reach zero. The protoypes found by the crisp and fuzzy algorithms are nearly identical, which is due to the fact that the real life dataset is a mixture of fuzzy and crisp labeled data points.

Comparing the fuzzy and crisp NPC and LVQ variants used in the experiments in terms of Fuzzy Cohen's Kappa $\widehat{\kappa}_F$ a perfect agreement of the classification accuracy for the training data and a substantial agreement for the test data was found.

4.2 Semi–supervised learning

Semi-supervised learning combines unsupervised and supervised learning. The most intuitive approach is simple post labeling after unsupervised training. The methods presented in the next two section are based on a modification of the cost functions of NG and HESKES' SOM. Thereby, an additional term, which takes the

classification error of the prototypes into account, is added to the respective cost functions. Since the labels of the data samples as well as the prototype labels are allowed to be fuzzy, the modified methods are called Fuzzy Labeled NG (FLNG) and Fuzzy Labeled SOM (FLSOM). Appropriate update rules for the prototypes and their class labels are derived in the following.

4.2.1 Fuzzy Labeled NG

In this section a modification of the unsupervised Neural Gas (NG) (Martinetz et al. 1993) and the unsupervised Batch Neural Gas (Cottrell et al. 2006) as described in section 2.1.4 are presented. The respective cost functions are modified to incorporate fuzzy labeled data and prototypes, in order to enable the algorithms to solve fuzzy classification tasks. Since it has to be distinguished between discrete and continuous data, different variants of the cost function together with appropriate update rules are provided. As a further aspect relevance learning is integrated into the approach to improve the classification results.

Cost function

To switch from the unsupervised NG to a supervised version the data vectors as well as the prototypes have to be equipped with labels reflecting their class assignments. As before, the fuzzy class labels are denoted as $\boldsymbol{c}_i$ for the data samples and $\boldsymbol{y}_j$ for the prototypes with possibilistic assignments $c_i^k \in [0,1]$ respectively $y_j^k \in [0,1]$ to the N_C classes.

To adapt the cost function, the original NG cost function is extended to incorporate fuzzy class labels. To preserve the excellent NG learning properties, the new cost function E_{FLNG} for fuzzy labeled data and prototypes is a balanced combination of the crisp NG cost function E_{NG} and an additional term E_{FL} to integrate the fuzzy label accuracy

$$E_{FLNG} = (1-\beta)E_{NG} + \beta E_{FL} \tag{4.23}$$

where the balance factor $\beta \in [0,1]$ weights the influence of the fuzziness.

Derivation of learning rules for discrete data

For discrete data $\boldsymbol{v}_i \in \mathbb{R}^d, i = 1,\ldots,N_V$ the FLNG cost function is a combination of the Batch NG (Cottrell et al. 2006) cost function (2.24) described in section 2.1.4

$$E_{NG} = \frac{1}{C(\sigma)} \sum_{j=1}^{N_P} \sum_{i=1}^{N_V} h_\sigma(k_j(\boldsymbol{v}_i, W)) \cdot d(\boldsymbol{v}_i, \boldsymbol{w}_j)$$

and an additional term for the fuzzy labels which can be defined as

$$E_{FL} = \frac{1}{2}\sum_{j=1}^{N_P}\sum_{i=1}^{N_V} h_\sigma(k_j(\boldsymbol{v}_i, W)) \cdot d_{FL}(\boldsymbol{c}_i, \boldsymbol{y}_j). \tag{4.24}$$

Thereby, d_{FL} is the classification error and usually the squared Euclidean distance is applied.

To obtain the learning rules, the derivatives with respect to the prototypes $\boldsymbol{w}_j$ and the fuzzy labels $\boldsymbol{y}_j$ have to be considered. The fuzzy label update is simply obtained as

$$\frac{\partial E_{FLNG}}{\partial \boldsymbol{y}_j} = (1-\beta)\frac{\partial E_{NG}}{\partial \boldsymbol{y}_j} + \beta\frac{\partial E_{FL}}{\partial \boldsymbol{y}_j} \tag{4.25}$$

with

$$\frac{\partial E_{NG}}{\partial \boldsymbol{y}_j} = 0 \tag{4.26}$$

$$\frac{\partial E_{FL}}{\partial \boldsymbol{y}_j} = \frac{1}{2}\sum_{i=1}^{N_V} h_\sigma\left(k_j(\boldsymbol{v}_i, W)\right)\frac{\partial d_{FL}(\boldsymbol{c}_i, \boldsymbol{y}_j)}{\partial \boldsymbol{y}_j} \tag{4.27}$$

which yields for $d_{FL}(\boldsymbol{c}_i, \boldsymbol{y}_j)$ set to the squared Euclidean distance

$$\Delta\boldsymbol{y}_j = -\sum_{i=1}^{N_V} h_\sigma\left(k_j(\boldsymbol{v}_i, W)\right)(\boldsymbol{c}_i - \boldsymbol{y}_j) \tag{4.28}$$

and is the weighted average of all fuzzy data labels. The factor β can be omitted, since there is no balancing necessary.

For the weight vector update the derivative of E_{FLNG} with respect to the prototypes $\boldsymbol{w}_j$

$$\frac{\partial E_{FLNG}}{\partial \boldsymbol{w}_j} = (1-\beta)\frac{\partial E_{NG}}{\partial \boldsymbol{w}_j} + \beta\frac{\partial E_{FL}}{\partial \boldsymbol{w}_j} \tag{4.29}$$

has to be considered. The first term $\frac{\partial E_{NG}}{\partial \boldsymbol{w}_j}$ is known from the original NG and yields the prototype update rule given in eq. (2.23)

$$\Delta\boldsymbol{w}_j^{NG} = -\varepsilon \cdot h_\sigma(k_j(\boldsymbol{v}_i, W)) \cdot \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j}.$$

Considering the second term yields

$$\frac{\partial E_{FL}}{\partial \boldsymbol{w}_j} = -\frac{1}{2\sigma}\sum_{r=1}^{N_P}\sum_{i=1}^{N_V}\frac{\partial k_r(\boldsymbol{v}_i, W)}{\partial \boldsymbol{w}_j} h_\sigma\left(k_r(\boldsymbol{v}_i, W)\right) d_E(\boldsymbol{c}_i, \boldsymbol{y}_r) \tag{4.30}$$

leaving the partial derivative of the rank function to solve. Yet, the rank function as defined in eq. (2.22) is not differentiable. Therefore, it has to be reformulated using the Heaviside function

$$k_r(\boldsymbol{v}_i, W) = \sum_{l=1}^{N_P} \theta\left(d(\boldsymbol{v}_i, \boldsymbol{w}_r) - d(\boldsymbol{v}_i, \boldsymbol{w}_l)\right) \tag{4.31}$$

for which the derivative is defined as the Dirac distribution $\frac{d\theta(x)}{dx} = \delta(x)$. Further, to simplify the notation the delta function $\Delta(\boldsymbol{v}_i, \boldsymbol{w}_j, \boldsymbol{w}_r) = d(\boldsymbol{v}_i, \boldsymbol{w}_j) - d(\boldsymbol{v}_i, \boldsymbol{w}_r)$ is introduced. Now the derivative $\frac{\partial k_r(\boldsymbol{v}_i, W)}{\partial \boldsymbol{w}_j}$ can be considered

$$\frac{\partial k_r(\boldsymbol{v}_i, W)}{\partial \boldsymbol{w}_j} = \delta_{r,j} \sum_{l=1}^{N_P} \delta\left(\Delta(\boldsymbol{v}_i, \boldsymbol{w}_r, \boldsymbol{w}_l)\right) \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_r)}{\partial \boldsymbol{w}_j} - \delta\left(\Delta(\boldsymbol{v}_i, \boldsymbol{w}_r, \boldsymbol{w}_j)\right) \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j}. \tag{4.32}$$

The Kronecker symbol $\delta_{r,j}$ causes the exclusion of those sum components, which are irrelevant concerning the derivative with respect to $\boldsymbol{w}_j$. Putting eq. (4.32) into eq. (4.30) finally yields

$$\begin{aligned} \frac{\partial E_{FL}}{\partial \boldsymbol{w}_j} \quad = \quad & -\frac{1}{2\sigma} \sum_{i=1}^{N_V} \left(\sum_{l=1}^{N_P} \delta\left(\Delta(\boldsymbol{v}_i, \boldsymbol{w}_j, \boldsymbol{w}_l)\right) \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j} \right) h_\sigma\left(k_j(\boldsymbol{v}_i, W)\right) d_E(\boldsymbol{c}_i, \boldsymbol{y}_j) \\ & + \frac{1}{2\sigma} \sum_{r=1}^{N_P} \sum_{i=1}^{N_V} \left(\delta\left(\Delta(\boldsymbol{v}_i, \boldsymbol{w}_r, \boldsymbol{w}_j)\right) \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j} \right) h_\sigma\left(k_r(\boldsymbol{v}_i, W)\right) d_E(\boldsymbol{c}_i, \boldsymbol{y}_r) \end{aligned} \tag{4.33}$$

Obviously, this term contributes to the weight update only for vanishing Δ function, i. e. on the borders of the receptive fields of the prototypes. Since in the case of discrete data the probability therefore is zero, it can be concluded, that the weight vector update based on the cost function (4.23) with E_{FL} as defined in eq. (4.24) is independent of the label adaption. Thus, the prototype update rule is identical with the update rule known from NG and can be stated as

$$\Delta \boldsymbol{w}_j = -\varepsilon \cdot h_\sigma(k_j(\boldsymbol{v}_i, W)) \cdot \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j}. \tag{4.34}$$

Again, the balancing factor β can be omitted.

For a distance $d^{\boldsymbol{\lambda}}(\boldsymbol{v}_i, \boldsymbol{w}_j)$ incorporating a relevance parameter $\boldsymbol{\lambda} \in \mathbb{R}^d$ as described in section 2.3.1 this parameter can be learned as well. The update rule is

derived by taking the derivative of cost function (4.23) with respect to the kth component of the relevance parameter $\boldsymbol{\lambda}$

$$\frac{\partial E_{FLNG}}{\partial \lambda_k} = (1-\beta)\frac{\partial E_{NG}}{\partial \lambda_k} + \beta \frac{\partial E_{FL}}{\partial \lambda_k} \tag{4.35}$$

where $\lambda_k \in [0,1]$ and $\sum_{k=1}^{d} \lambda_k = 1$. Considering the first term yields

$$\begin{aligned} \frac{\partial E_{NG}}{\partial \lambda_k} &= \frac{1}{C(\sigma)} \sum_{j=1}^{N_P} \sum_{i=1}^{N_V} \Big[h_\sigma \left(k_j(\boldsymbol{v}_i, W)\right) \cdot \frac{\partial k_j(\boldsymbol{v}_i, W)}{\partial \lambda_k} \cdot d^\lambda(\boldsymbol{v}_i, \boldsymbol{w}_j) \\ &\quad + h_\sigma \left(k_j(\boldsymbol{v}_i, W)\right) \cdot \frac{\partial d^\lambda(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \lambda_k} \Big]. \end{aligned} \tag{4.36}$$

The derivative of the rank function $k_j(\boldsymbol{v}_i, W)$ with respect to λ_k is obtained as

$$\frac{\partial k_j(\boldsymbol{v}_i, W)}{\partial \lambda_k} = \sum_{l=1}^{N_P} \delta(\Delta_\lambda(\boldsymbol{v}_i, \boldsymbol{w}_j, \boldsymbol{w}_l) \cdot \frac{\partial \Delta_\lambda(\boldsymbol{v}_i, \boldsymbol{w}_j, \boldsymbol{w}_l)}{\partial \lambda_k} \tag{4.37}$$

where $\Delta_\lambda(\boldsymbol{v}_i, \boldsymbol{w}_j, \boldsymbol{w}_l) = d^\lambda(\boldsymbol{v}_i, \boldsymbol{w}_j) - d^\lambda(\boldsymbol{v}_i, \boldsymbol{w}_l)$. Since δ is symmetric, eq. (4.37) is non-vanishing only for $d^\lambda(\boldsymbol{v}_i, \boldsymbol{w}_j) = d^\lambda(\boldsymbol{v}_i, \boldsymbol{w}_l)$ and therefore eq. (4.36) can be rewritten as

$$\frac{\partial E_{NG}}{\partial \lambda_k} = \frac{1}{C(\sigma)} \sum_{j=1}^{N_P} \sum_{i=1}^{N_V} h_\sigma \left(k_j(\boldsymbol{v}_i, W)\right) \cdot \frac{\partial d^\lambda(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \lambda_k}. \tag{4.38}$$

For the same reason the second term of eq. (4.35) needs not to be considered and the relevance parameter update rule can be stated as

$$\Delta\lambda_k = -\varepsilon_\lambda \sum_{j=1}^{N_P} h_\sigma \left(k_j(\boldsymbol{v}_i, W)\right) \cdot \frac{\partial d^\lambda(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \lambda_k}. \tag{4.39}$$

Derivation of learning rules for continuous data

In case of continous data, denoted as $\boldsymbol{v}$ with class labels $\boldsymbol{c}$, the cost function known from the original NG (Martinetz et al. 1993) and given in eq. (2.21) has to be considered

$$E_{NG} = \frac{1}{2C(\sigma)} \sum_{j=1}^{N_P} \int h_\sigma(k_j(\boldsymbol{v}, W)) \cdot d(\boldsymbol{v}, \boldsymbol{w}_j) P(V) d\boldsymbol{v}.$$

As stated in section 2.1.4 the neighborhood function h_σ is a Gaussian shaped curve $h_\sigma(t) = \exp(-t/\sigma)$ with the rank function $k_j(\boldsymbol{v}_i, W) = |\{\boldsymbol{w}_k | d(\boldsymbol{v}_i, \boldsymbol{w}_k) < d(\boldsymbol{v}_i, \boldsymbol{w}_j)\}|$

as parameter and neighborhood range $\sigma > 0$. Yet, contrary to the derivation of the update rules for discrete data, the borders of the receptive fields cannot be ignored and the above argument, concluding that the prototype update is independent of the label adaption, is not valid. Therefore, the problem has to be treated differently and the term E_{FL} of the cost function has to be redefined. In the following two possible redefinitions are presented. The first solution applies a Gaussian kernel to weight the label error and the second approximates the rank function.

Gaussian kernel

The first method weights the label error by a Gaussian kernel depending on the distance. The term E_{FL} is chosen as

$$E_{FL} = \frac{1}{2}\sum_{j=1}^{N_P}\int P(\boldsymbol{v})g_\gamma(\boldsymbol{v},\boldsymbol{w}_j)d_{FL}(\boldsymbol{c},\boldsymbol{y}_j)d\boldsymbol{v} \tag{4.40}$$

where

$$g_\gamma(\boldsymbol{v}_i,\boldsymbol{w}_j) = \exp\left(-\frac{d(\boldsymbol{v}_i,\boldsymbol{w}_j)}{2\gamma^2}\right) \tag{4.41}$$

is a Gaussian kernel describing a neighborhood range in the data space. Since $g_\gamma(\boldsymbol{v}_i,\boldsymbol{w}_j)$ depends on the prototype locations, E_{FL} is influenced by the prototypes $\boldsymbol{w}_j$ as well as their labels $\boldsymbol{y}_j$. To obtain the update rules, again, the derivative of the cost function with respect to $\boldsymbol{w}_j$ and $\boldsymbol{y}_j$ has to be considered. For the prototype update

$$\frac{\partial E_{FLNG}}{\partial \boldsymbol{w}_j} = (1-\beta)\frac{\partial E_{NG}}{\partial \boldsymbol{w}_j} + \beta\frac{\partial E_{FL}}{\partial \boldsymbol{w}_j} \tag{4.42}$$

the first term $\frac{\partial E_{NG}}{\partial \boldsymbol{w}_j}$ is known from the original NG and yields eq. (2.23)

$$\Delta\boldsymbol{w}_j^{NG} = -\varepsilon\cdot h_\sigma(k_j(\boldsymbol{v}_i,W))\cdot\frac{\partial d(\boldsymbol{v}_i,\boldsymbol{w}_j)}{\partial \boldsymbol{w}_j}.$$

The second term $\frac{\partial E_{FL}}{\partial \boldsymbol{w}_j}$ now contributes acording to

$$\frac{\partial E_{FL}}{\partial \boldsymbol{w}_j} = -\frac{1}{4\gamma^2}\int P(\boldsymbol{v})g_\gamma(\boldsymbol{v},\boldsymbol{w}_j)\frac{\partial d(\boldsymbol{v},\boldsymbol{w}_j)}{\partial \boldsymbol{w}_j}d_{FL}(\boldsymbol{c},\boldsymbol{y}_j)d\boldsymbol{v} \tag{4.43}$$

which takes the accuracy of fuzzy labeling into account for the weight update. Both terms define the learning rule for the weights

$$\Delta\boldsymbol{w}_j = -\varepsilon_1\cdot(1-\beta)\cdot h_\sigma(k_j(\boldsymbol{v}_i,W))\frac{\partial d(\boldsymbol{v}_i,\boldsymbol{w}_j)}{\partial \boldsymbol{w}_j} - \varepsilon_2\cdot\frac{\beta}{4\gamma^2}\cdot g_\gamma(\boldsymbol{v}_i,\boldsymbol{w}_j)\frac{\partial d(\boldsymbol{v}_i,\boldsymbol{w}_j)}{\partial \boldsymbol{w}_j}(\boldsymbol{c}_i-\boldsymbol{y}_j)^2.$$

(4.44)

For the fuzzy labels simply $\frac{\partial E_{FLNG}}{\partial \boldsymbol{y}_j} = \frac{\partial E_{FL}}{\partial \boldsymbol{y}_j}$ is obtained, where

$$\frac{\partial E_{FL}}{\partial \boldsymbol{y}_j} = \frac{1}{2} \int P(\boldsymbol{v}) g_\gamma(\boldsymbol{v}, \boldsymbol{w}_j) \frac{\partial d_{FL}(\boldsymbol{c}, \boldsymbol{y}_j)}{\partial \boldsymbol{y}_j} d\boldsymbol{v} \tag{4.45}$$

which is the weighted average of the fuzzy data labels of those data belonging to the receptive field of the associated prototypes. However, in comparison to the original NG (Martinetz et al. 1993) the receptive fields are different due to the modified learning rule for the prototypes and their resulting different locations. The learning rule given a data point $\boldsymbol{v}$ yields for the squared Euclidean distance $d_{FL}(\boldsymbol{c}_i, \boldsymbol{y}_j)$

$$\Delta \boldsymbol{y}_j = \varepsilon_y g_\gamma(\boldsymbol{v}_i, \boldsymbol{w}_j)(\boldsymbol{c}_i - \boldsymbol{y}_j) \tag{4.46}$$

where the balancing factor β is not required.

Considering a general metric $\delta^\lambda(\boldsymbol{v}_i, \boldsymbol{w}_j)$ the update rule for the relevance parameter $\boldsymbol{\lambda}$ can be derived by stochastic gradient descent of the cost function

$$\frac{\partial E_{FLNG}}{\partial \lambda_k} = (1-\beta)\frac{\partial E_{NG}}{\partial \lambda_k} + \beta \frac{\partial E_{FL}}{\partial \lambda_k}. \tag{4.47}$$

The first term $\frac{\partial E_{NG}}{\partial \lambda_k}$ gives

$$\begin{aligned} \frac{\partial E_{NG}}{\partial \lambda_k} &= \frac{1}{C(\sigma)} \left[\sum_{j=1}^{N_P} \int P(\boldsymbol{v}) h_\sigma\left(k_j(\boldsymbol{v}, W)\right) \cdot \frac{\partial d^\lambda(\boldsymbol{v}, \boldsymbol{w}_j)}{\partial \lambda_k} d\boldsymbol{v} \right. \\ &\qquad \left. + \sum_{j=1}^{N_P} \int P(\boldsymbol{v}) d^\lambda(\boldsymbol{v}, \boldsymbol{w}_j) \frac{\partial h_\sigma\left(k_j(\boldsymbol{v}, W)\right)}{\partial \lambda_k} d\boldsymbol{v} \right] \\ &= \frac{1}{C(\sigma)} \sum_{j=1}^{N_P} \int P(\boldsymbol{v}) h_\sigma\left(k_j(\boldsymbol{v}, W)\right) \cdot \frac{\partial d^\lambda(\boldsymbol{v}, \boldsymbol{w}_j)}{\partial \lambda_k} d\boldsymbol{v} \end{aligned} \tag{4.48}$$

following the argument stated above eq. (4.38). The second term $\frac{\partial E_{FL}}{\partial \lambda_k}$ yields

$$\frac{\partial E_{FL}}{\partial \lambda_k} = -\frac{1}{4\gamma^2} \sum_{j=1}^{N_P} \int P(\boldsymbol{v}) g_\gamma(\boldsymbol{v}, \boldsymbol{w}_j) \cdot \frac{\partial d^\lambda(\boldsymbol{v}, \boldsymbol{w}_j)}{\partial \lambda_k} \cdot d_E(\boldsymbol{c}_i, \boldsymbol{y}_j) d\boldsymbol{v}. \tag{4.49}$$

Combining eqs. (4.48) and (4.49) results in the relevance parameter update rule

$$\begin{aligned} \Delta \lambda_k &= -\sum_{j=1}^{N_P} \frac{\partial d^\lambda(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \lambda_k} \left[\varepsilon_{\lambda_1} \frac{1-\beta}{C(\sigma)} h_\sigma\left(k_i(\boldsymbol{v}_i, W)\right) \right. \\ &\qquad \left. + \varepsilon_{\lambda_2} \frac{\beta}{4\gamma^2} g_\gamma(\boldsymbol{v}_i, \boldsymbol{w}_j) d_{FL}(\boldsymbol{c}_i, \boldsymbol{y}_j) \right]. \end{aligned} \tag{4.50}$$

Approximation of the rank function

In the second approach the neighborhood function h_σ is approximated. Therefore the rank function eq. (2.22) respectively the formulation as Heaviside function eq. (4.31) is replaced by a sigmoid function $\zeta(x) = \left(1 + \exp\left(-\frac{x}{2\tau^2}\right)\right)^{-1}$. This way an approximation of the rank is obatined:

$$\begin{aligned} \widetilde{k}_j(\boldsymbol{v}_i, W) &= \sum_{j=1}^{N_P} \zeta\left(d(\boldsymbol{v}_i, \boldsymbol{w}_j) - d(\boldsymbol{v}_i, \boldsymbol{w}_l)\right) \\ &= \sum_{j=1}^{N_P} \zeta\left(\Delta(\boldsymbol{v}_i, \boldsymbol{w}_j, \boldsymbol{w}_l)\right) \end{aligned} \tag{4.51}$$

with $\Delta(\boldsymbol{v}_i, \boldsymbol{w}_j, \boldsymbol{w}_l) = (d(\boldsymbol{v}_i, \boldsymbol{w}_j) - d(\boldsymbol{v}_i, \boldsymbol{w}_l))$ as introduced on page 92. Now, the second term of the cost function (4.23) regarding the fuzzy labels can be defined as

$$E_{FL} = \frac{1}{2} \sum_{j=1}^{N_P} \int P(\boldsymbol{v}) \widetilde{h}_\sigma \left(\widetilde{k}_j(\boldsymbol{v}, W)\right) d_{FL}(\boldsymbol{c}, \boldsymbol{y}_j) d\boldsymbol{v} \tag{4.52}$$

with $\widetilde{h}_\sigma = \exp\left(-\frac{\widetilde{k}_j(\boldsymbol{v}, W)}{\sigma}\right)$. To obtain the update rules, again the derivative of E_{FLNG} with respect to $\boldsymbol{w}_j$ and $\boldsymbol{y}_j$ has to be considered. The update rule for the fuzzy labels is simply obtained as

$$\frac{\partial E_{FLNG}}{\partial \boldsymbol{y}_j} = \frac{1}{2} \int P(\boldsymbol{v}) \widetilde{h}_\sigma \left(\widetilde{k}_j(\boldsymbol{v}, W)\right) \frac{\partial d_E(\boldsymbol{c}, \boldsymbol{y}_j)}{\partial \boldsymbol{y}_j} d\boldsymbol{v} \tag{4.53}$$

which is the weighted average of all fuzzy labels of the data. Given a data point $\boldsymbol{v}$ and solving for the squared Euclidean distance $d_{FL}(\boldsymbol{c}, \boldsymbol{y}_j)$ the update rule omitting the balance factor β yields

$$\Delta \boldsymbol{y}_j = -\varepsilon_y \widetilde{h}_\sigma \left(\widetilde{k}_j(\boldsymbol{v}_i, W)\right) (\boldsymbol{c}_i - \boldsymbol{y}_j). \tag{4.54}$$

For the weight vector update the gradient $\frac{\partial E_{FLNG}}{\partial \boldsymbol{w}_j}$ has to be considered. The first term $\frac{\partial E_{NG}}{\partial \boldsymbol{w}_j}$ is known from the original NG (2.23). The second term including the fuzzy label error yields

$$\frac{\partial \widetilde{E}_{FL}}{\partial \boldsymbol{w}_j} = -\frac{1}{2\sigma} \sum_{j=1}^{N_P} \int P(\boldsymbol{v}) \frac{\partial \widetilde{k}_j(\boldsymbol{v}, W)}{\partial \boldsymbol{w}_j} \widetilde{h}_\sigma \left(\widetilde{k}_j(\boldsymbol{v}, W)\right) d_{FL}(\boldsymbol{c}, \boldsymbol{y}_j) d\boldsymbol{v} \tag{4.55}$$

where the derivative of the rank funktion $\widetilde{k}_j(\boldsymbol{v}, W)$ with respect to $\boldsymbol{w}_j$ yields

$$\frac{\partial \widetilde{k}_j(\boldsymbol{v}_i, W)}{\partial \boldsymbol{w}_j} = \delta_{r,j} \cdot \left(\sum_{l=1}^{N_P} \zeta'\left(\Delta(\boldsymbol{v}_i, \boldsymbol{w}_j, \boldsymbol{w}_l)\right) \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_r)}{\partial \boldsymbol{w}_j} \right) - \zeta'\left(\Delta(\boldsymbol{v}_i, \boldsymbol{w}_r, \boldsymbol{w}_j)\right) \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j}$$

(4.56)

with $\zeta'(x) = \frac{1}{2\tau^2}\zeta(x)(1-\zeta(x))$. In combination with eq. (4.55) is obtained

$$\begin{aligned}\frac{\partial \widetilde{E}_{FL}}{\partial \boldsymbol{w}_j} &= -\frac{1}{2\sigma}\int P(\boldsymbol{v})\left(\sum_{l=1}^{N_P}\zeta'\left(\Delta(\boldsymbol{v},\boldsymbol{w}_j,\boldsymbol{w}_l)\right)\frac{\partial d(\boldsymbol{v},\boldsymbol{w}_j)}{\partial \boldsymbol{w}_j}\right)\widetilde{h}_\sigma\left(\widetilde{k}_j(\boldsymbol{v},W)\right)d_E(\boldsymbol{c},\boldsymbol{y}_j)d\boldsymbol{v} \\ &\quad +\frac{1}{2\sigma}\sum_{r=1}^{N_P}\int P(\boldsymbol{v})\left(\zeta'\left(\Delta(\boldsymbol{v},\boldsymbol{w}_r,\boldsymbol{w}_j)\right)\frac{\partial d(\boldsymbol{v},\boldsymbol{w}_j)}{\partial \boldsymbol{w}_j}\right)\widetilde{h}_\sigma\left(\widetilde{k}_r(\boldsymbol{v},W)\right)d_E(\boldsymbol{c},\boldsymbol{y}_r)d\boldsymbol{v}.\end{aligned} \tag{4.57}$$

The full prototype update rule can now be formulated as

$$\begin{aligned}\Delta\boldsymbol{w}_j &= -\varepsilon_1\cdot(1-\beta)\cdot h_\sigma\left(k_j(\boldsymbol{v}_i,W)\right)\cdot\frac{\partial d(\boldsymbol{v}_i,\boldsymbol{w}_j)}{\partial \boldsymbol{w}_j} \\ &\quad -\varepsilon_2\cdot\frac{\beta}{\sigma}\left[\widetilde{h}_\sigma\left(\widetilde{k}_j(\boldsymbol{v}_i,W)\right)d_{FL}(\boldsymbol{c}_i,\boldsymbol{y}_j)\sum_{l=1}^{N_P}\zeta'\left(\Delta(\boldsymbol{v}_i,\boldsymbol{w}_j,\boldsymbol{w}_l)\right)\right. \\ &\quad \left.-\sum_{r=1}^{N_P}\widetilde{h}_\sigma\left(\widetilde{k}_r(\boldsymbol{v}_i,W)\right)\zeta'\left(\Delta(\boldsymbol{v}_i,\boldsymbol{w}_r,\boldsymbol{w}_j)\right)d_{FL}(\boldsymbol{c}_i,\boldsymbol{y}_r)\right]\frac{\partial d(\boldsymbol{v}_i,\boldsymbol{w}_j)}{\partial \boldsymbol{w}_j}\end{aligned} \tag{4.58}$$

Finally, considering a general metric $d^\lambda(\boldsymbol{v},\boldsymbol{w}_j)$ with a relevance parameter $\boldsymbol{\lambda}$, the update rule thereof can again be obtained by stochastic gradient descent. The first term $\frac{\partial E_{NG}}{\partial \lambda_k}$ is identical with eq. (4.48) since only the term concerning the fuzzy labels E_{FL} is changed. The partial derivative of E_{FL} with respect to λ_k yields

$$\frac{\partial E_{FL}}{\partial \lambda_k} = -\frac{1}{2}\sum_{j=1}^{N_P}\int P(\boldsymbol{v})\cdot\widetilde{h}_\sigma\left(\widetilde{k}_j(\boldsymbol{v},W)\right)\cdot\frac{\partial \widetilde{k}_j(\boldsymbol{v},W)}{\partial \lambda_k}\cdot d_{FL}(\boldsymbol{c},\boldsymbol{y}_j)d\boldsymbol{v} \tag{4.59}$$

with $\frac{\partial \widetilde{k}_j(\boldsymbol{v},W)}{\partial \lambda_k} = \sum_{l=1}^{N_P}\zeta'\left(\Delta_\lambda(\boldsymbol{v},\boldsymbol{w}_j,\boldsymbol{w}_l)\right)\frac{\partial \Delta_\lambda(\boldsymbol{v},\boldsymbol{w}_j,\boldsymbol{w}_l)}{\partial \lambda_k}$. By plugging eqs. (4.48) and (4.59) into eq. (4.47) the update rule for the relevance parameter is obtained as

$$\begin{aligned}\Delta\lambda_k &= -\varepsilon_{\lambda_1}\frac{(1-\beta)}{C(\sigma)}\sum_{j=1}^{N_P}h_\sigma\left(k_j(\boldsymbol{v}_i,\boldsymbol{w}_j)\right)\cdot\frac{\partial d^\lambda(\boldsymbol{v}_i,\boldsymbol{w}_j)}{\partial \lambda_k} \\ &\quad -\varepsilon_{\lambda_2}\frac{\beta}{2}\sum_{j=1}^{N_P}\widetilde{h}_\sigma\left(\widetilde{k}_j(\boldsymbol{v}_i,\boldsymbol{w}_j)\right)\cdot\frac{\partial \widetilde{k}_j(\boldsymbol{v}_i,W)}{\partial \lambda_k}\cdot d_{FL}(\boldsymbol{c}_i,\boldsymbol{y}_j).\end{aligned} \tag{4.60}$$

Experiments analyzing the behavior and performance of the above algorithm in all its modifications can be found in (Villmann et al. 2005).

4.2.2 Fuzzy Labeled SOM

Self Organizing Maps (SOM) introduced by KOHONEN (Kohonen 1990) and described in detail in section 2.1.3 employ an unsupervised learning scheme based on prototypes positioned on a fixed grid. Thereby prototypes representing similar data occopy positions close to each other on the grid. If data labels are available they might be used for posterior labeling. But clustering high-dimensional data often results in a suboptimal solution. An algorithm incorporating existing data labels during the adaption process allows supervised learning with improved classification. Further visualization by means of the underlying SOM topology will be improved due to the now label adapted topology.

The integration of label information requires a cost function to perform a supervised learning based on a stochastic gradient descent. HESKES proposed a modification of the original SOM which exhibits such a cost function (Heskes 1999). The following consideration are based on this formulation, see also Section 2.1.3.

Furthermore, since the derivation is based on a general distance measure, the algorithm can easily be adapted to incorporate metric learning as known from learning vector quantization (Hammer and Villmann 2002, Hammer et al. 2005). This way, the flexibility of the approach can be improved and better clustering results be achieved.

Cost function

As before, each training sample $\boldsymbol{v}_i$ is associated with a N_C-dimensional fuzzy class label $\boldsymbol{c}_i$, where the components $c_i^k \in [0,1]$ denote the assignment of data point $\boldsymbol{v}_i$ to class $k = 1, \ldots, N_C$. The class memberships can either be possibilistic or probabilistic. Accordingly, the labels for the prototypes $\boldsymbol{w}_j$ are denoted as $\boldsymbol{y}_j$ with $y_j^k \in [0,1]$ and determine the portion to which neuron j is assigned to the respective class k.

Since during the training the prototypes as well as their class assignments are adapted, the cost function has to be modified to integrate the fuzzy class memberships. For this purpose the SOM cost function as defined in (2.20) is extended by an additional term to obtain the cost function for fuzzy labeled data (FLSOM):

$$E_{FLSOM} = (1-\beta)E_{SOM} + \beta E_{FL} \tag{4.61}$$

The factor $\beta \in [0,1]$ is a balance factor to regulate the impact of the fuzzy label learning. A simple choice is a fixed value of $\beta = 0.5$, but depending on the main goal – clustering or classification – this factor could also be changed during training to shift the focus. While the first term E_{SOM} (2.20) measures the quantization of the map taking topological constraints into account, the second term E_{FL} measures the

error of the classification. The SOM cost function (2.20) is given as

$$E_{SOM} = \frac{1}{C(\sigma)} \int P(\boldsymbol{v}) \sum_{j=1}^{N_P} \delta_j^{s(\boldsymbol{v})} \sum_{l=1}^{N_P} h_\sigma(j,l) \cdot d(\boldsymbol{v}, \boldsymbol{w}_l) d\boldsymbol{v}.$$

The classification error E_{FL} can be defined as

$$E_{FL} = \int P(\boldsymbol{v}) \sum_{j=1}^{N_P} g_\gamma(\boldsymbol{v}, \boldsymbol{w}_j) d_{FL}(\boldsymbol{c}, \boldsymbol{y}_j) d\boldsymbol{v} \tag{4.62}$$

where $d_{FL}(\boldsymbol{c}, \boldsymbol{y}_j)$ is a measure for the dissimilarity between the class labels of prototype $\boldsymbol{w}_j$ respectively neuron j and data point $\boldsymbol{v}_i$. The common choice for this distance is the squared Euclidean distance, but depending on the specific task any other continuous and differentiable measure might be suitable. The neighborhood range in the data space is described by a Gaussian kernel

$$g_\gamma(\boldsymbol{v}, \boldsymbol{w}_j) = \exp\left(-\frac{d(\boldsymbol{v}, \boldsymbol{w}_j)}{2\gamma^2}\right). \tag{4.63}$$

This choice is based on the assumption that data points close to the prototype determine the corresponding label if the underlying classification is sufficiently smooth. Note that $g_\gamma(\boldsymbol{v}, \boldsymbol{w}_j)$ depends on the prototype locations, such that E_{FL} is influenced by both $\boldsymbol{w}_j$ and $\boldsymbol{y}_j$.

Derivation of learning rules

The update rules for the prototypes $\boldsymbol{w}_j$ and their class labels $\boldsymbol{y}_j$ are obtained by stochastic gradient descent on the cost function. The derivative with respect to the the prototypes yields

$$\frac{\partial E_{FLSOM}}{\partial \boldsymbol{w}_j} = (1-\beta)\frac{\partial E_{SOM}}{\partial \boldsymbol{w}_j} + \beta \frac{\partial E_{FL}}{\partial \boldsymbol{w}_j} \tag{4.64}$$

with

$$\frac{\partial E_{SOM}}{\partial \boldsymbol{w}_j} = \frac{1}{C(\sigma)} \int P(\boldsymbol{v}) h_\sigma(j, s(\boldsymbol{v})) \frac{\partial d(\boldsymbol{v}, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j} d\boldsymbol{v} \tag{4.65}$$

$$\frac{\partial E_{FL}}{\partial \boldsymbol{w}_j} = -\frac{1}{2\gamma^2} \int P(\boldsymbol{v}) g_\gamma(\boldsymbol{v}, \boldsymbol{w}_j) \frac{\partial d(\boldsymbol{v}, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j} d_{FL}(\boldsymbol{c}, \boldsymbol{y}_j) d\boldsymbol{v} \tag{4.66}$$

$$\tag{4.67}$$

and therefore the update rule can be formulated as

$$\begin{aligned}\Delta \boldsymbol{w}_j &= -\varepsilon_1 \cdot \frac{(1-\beta)}{C(\sigma)} \cdot h_\sigma(j, s(\boldsymbol{v}_i)) \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j} \\ &\quad + \varepsilon_2 \cdot \frac{\beta}{2\gamma^2} \cdot g_\gamma(\boldsymbol{v}_i, \boldsymbol{w}_j) \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j} d_{FL}(\boldsymbol{c}_i, \boldsymbol{y}_j) \\ &= -\left(\varepsilon_1 \cdot \frac{(1-\beta)}{C(\sigma)} h_\sigma(j, s(\boldsymbol{v}_i)) - \varepsilon_2 \frac{\beta}{2\gamma^2} \cdot g_\gamma(\boldsymbol{v}_i, \boldsymbol{w}_j) d_{FL}(\boldsymbol{c}_i, \boldsymbol{y}_j) \right) \frac{\partial d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j}\end{aligned} \tag{4.68}$$

Since the prototype update for FLSOM also depends on the fuzzy labels, the receptive fields compared to the standard SOM are different. Dependig on the strength of the label the field might bend more or less to one or the other prototype.

The update rules for the fuzzy labels can be obtained analogously

$$\frac{\partial E_{FLSOM}}{\partial \boldsymbol{y}_j} = (1-\beta) \frac{\partial E_{SOM}}{\partial \boldsymbol{y}_j} + \beta \frac{\partial E_{FL}}{\partial \boldsymbol{y}_j} \tag{4.69}$$

$$\frac{\partial E_{SOM}}{\partial \boldsymbol{y}_j} = 0 \tag{4.70}$$

$$\frac{\partial E_{FL}}{\partial \boldsymbol{y}_j} = \int P(\boldsymbol{v}) g_\gamma(\boldsymbol{v}, \boldsymbol{w}_j) \frac{\partial d_{FL}(\boldsymbol{c}, \boldsymbol{y}_j)}{\partial \boldsymbol{y}_j} d\boldsymbol{v} \tag{4.71}$$

and the corresponding learning rule yields

$$\Delta \boldsymbol{y}_j = \varepsilon_{FL} \cdot \beta \cdot g_\gamma(\boldsymbol{v}_i, \boldsymbol{w}_j) \frac{\partial d_{FL}(\boldsymbol{c}_i, \boldsymbol{y}_j)}{\partial \boldsymbol{y}_j} \tag{4.72}$$

where $\varepsilon_{FL} > 0$ is the learning rate for the fuzzy prototype labels, which are influenced by the average of the data labels located close by.

As mentioned above, unsupervised SOMs generate a topographic mapping from the data space onto the prototype grid under specific conditions. If the classes are consistently determined with respect to the varying data, one can expect for supervised topographic FLSOMs that the labels become ordered within the grid structure of the prototype lattice. In this case, the topological order of the prototypes should be transferred to the topological order of prototype labels such that a smooth change of the fuzzy probabilistic class labels between neighbored grid positions is achieved.

Relevance learning

Incorporating relevance learning into the approach, the distance $d^\lambda(\boldsymbol{v}, \boldsymbol{w}_j)$ with a general parameter $\boldsymbol{\lambda}$ to weight the input dimensions has to be considered. The optimization of the cost function follows a stochastic gradient descend with respect to

the metric parameter $\boldsymbol{\lambda}$. The formal derivation yields

$$\frac{\partial E_{FLSOM}}{\partial \boldsymbol{\lambda}} = (1-\beta)\frac{\partial E_{SOM}}{\partial \boldsymbol{\lambda}} + \beta\frac{\partial E_{FL}}{\partial \boldsymbol{\lambda}} \tag{4.73}$$

with

$$\frac{\partial E_{SOM}}{\partial \boldsymbol{\lambda}} = \frac{1}{C(\sigma)} \int P(\boldsymbol{v}) \sum_{j=1}^{N_P} \delta_j^{s(\boldsymbol{v})} \sum_{l=1}^{N_P} h_\sigma(j,l) \frac{\partial d^\lambda(\boldsymbol{v}, \boldsymbol{w}_j)}{\partial \boldsymbol{\lambda}} d\boldsymbol{v} \tag{4.74}$$

$$\frac{\partial E_{FL}}{\partial \boldsymbol{\lambda}} = -\frac{1}{2\gamma^2} \int P(\boldsymbol{v}) \sum_{j=1}^{N_P} g_\gamma(\boldsymbol{v}, \boldsymbol{w}_j) \frac{\partial d^\lambda(\boldsymbol{v}, \boldsymbol{w}_j)}{\partial \boldsymbol{\lambda}} d_{FL}(\boldsymbol{c}, \boldsymbol{y}_j) d\boldsymbol{v}. \tag{4.75}$$

Note that the distance $d_{FL}(\boldsymbol{c}, \boldsymbol{y}_j)$ measuring the similarity of the labels is independent of the relevance parameter weighting the input dimension. The update rule can now be specified as

$$\begin{aligned} \Delta\boldsymbol{\lambda} = & -\varepsilon_{\lambda 1} \cdot \frac{1-\beta}{C(\sigma)} \cdot \sum_{j=1}^{N_P} h_\sigma(s(\boldsymbol{v}_i), j) \frac{\partial d^\lambda(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{\lambda}} \\ & + \varepsilon_{\lambda 2} \cdot \frac{\beta}{2\gamma^2} \cdot \sum_{j=1}^{N_P} g_\gamma(\boldsymbol{v}_i, \boldsymbol{w}_j) \frac{\partial d^\lambda(\boldsymbol{v}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{\lambda}} d_{FL}(\boldsymbol{c}_i, \boldsymbol{y}_j) \end{aligned} \tag{4.76}$$

where $\varepsilon_\lambda > 0$ is the learning rate for the relevance parameter update. The update itself has to be followed by normalization to ensure that the constraints on $\boldsymbol{\lambda}$ are kept.

Since this section is only intended to provide the theoretical derivation of Fuzzy Labeled SOM, refer to (Villmann, Seiffert, Schleif, Brüß, Geweniger and Hammer 2006) where the performance of this algorithm is examined in detail.

Appendices

4.A Derivation of the F-FSNPC window rule

In complete analogy to FSNPC in section 2.2.6, a window rule for the active region for the F-FSNPC prototype update can be derived. By setting

$$T = P(j|\boldsymbol{v})((\boldsymbol{c}_i - \boldsymbol{y}_j)^2 - lc_i)$$

and using the Gaussian form (2.65) for $P(j|\boldsymbol{v})$, the following steps lead to the term $T = T_0 \cdot \Pi(\boldsymbol{v}_i, \boldsymbol{c}_i, \boldsymbol{w}_j, \boldsymbol{y}_j)$, where T_0 defines the window rule.

$$\begin{aligned}
T &= \frac{\sum_k \left((\boldsymbol{c}_i - \boldsymbol{y}_j)^2 + (\boldsymbol{c}_i - \boldsymbol{y}_k)^2\right) e^{\left(-\frac{d(\boldsymbol{v}_i, \boldsymbol{w}_k}{2\sigma^2}\right)}}{\sum_k \left((\boldsymbol{c}_i - \boldsymbol{y}_j)^2 + (\boldsymbol{c}_i - \boldsymbol{y}_k)^2\right) e^{\left(-\frac{d(\boldsymbol{v}_i, \boldsymbol{w}_k}{2\sigma^2}\right)}} \cdot \frac{e^{\left(\frac{-d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{2\sigma^2}\right)}}{\sum_k e^{\left(\frac{-d(\boldsymbol{v}_i, \boldsymbol{w}_k)}{2\sigma^2}\right)}} \cdot ((\boldsymbol{c}_i - \boldsymbol{y}_j)^2 - lc_i) \\
&= \underbrace{\frac{\sum_k \left((\boldsymbol{c}_i - \boldsymbol{y}_j)^2 + (\boldsymbol{c}_i - \boldsymbol{y}_k)^2\right) e^{\left(-\frac{d(\boldsymbol{v}_i, \boldsymbol{w}_k}{2\sigma^2}\right)}}{\sum_k e^{\left(\frac{-d(\boldsymbol{v}_i, \boldsymbol{w}_k)}{2\sigma^2}\right)}}}_{a} \cdots \\
&\quad \cdots \underbrace{\frac{e^{\left(\frac{-d(\boldsymbol{v}_i, \boldsymbol{w}_j)}{2\sigma^2}\right)}}{\sum_k \left((\boldsymbol{c}_i - \boldsymbol{y}_j)^2 + (\boldsymbol{c}_i - \boldsymbol{y}_k)^2\right) e^{\left(-\frac{d(\boldsymbol{v}_i, \boldsymbol{w}_k}{2\sigma^2}\right)}}}_{\Pi(\boldsymbol{v}_i, \boldsymbol{c}_i, \boldsymbol{w}_j, \boldsymbol{y}_j)} \cdot ((\boldsymbol{c}_i - \boldsymbol{y}_j)^2 - lc_i) \\
&= a \cdot \Pi(\boldsymbol{v}_i, \boldsymbol{c}_i, \boldsymbol{w}_j, \boldsymbol{y}_j) \cdot ((\boldsymbol{c}_i - \boldsymbol{y}_j)^2 - lc_i) \qquad (4.77)
\end{aligned}$$

Considering the first term separately yields

$$
\begin{aligned}
a &= \frac{\sum_k \left((\boldsymbol{c}_i - \boldsymbol{y}_j)^2 + (\boldsymbol{c}_i - \boldsymbol{y}_k)^2\right) e^{\left(-\frac{d(\boldsymbol{v}_i, \boldsymbol{w}_k}{2\sigma^2}\right)}}{\sum_k e^{\left(\frac{-d(\boldsymbol{v}_i, \boldsymbol{w}_k)}{2\sigma^2}\right)}} \\
&= \sum_k \left((\boldsymbol{c}_i - \boldsymbol{y}_j)^2 + (\boldsymbol{c}_i - \boldsymbol{y}_k)^2\right) \frac{e^{\left(-\frac{d(\boldsymbol{v}_i, \boldsymbol{w}_k}{2\sigma^2}\right)}}{\sum_{k'} e^{\left(\frac{-d(\boldsymbol{v}_i, \boldsymbol{w}_{k'})}{2\sigma^2}\right)}} \\
&= \sum_k (\boldsymbol{c}_i - \boldsymbol{y}_j)^2 \underbrace{\frac{e^{\left(-\frac{d(\boldsymbol{v}_i, \boldsymbol{w}_k}{2\sigma^2}\right)}}{\sum_{k'} e^{\left(\frac{-d(\boldsymbol{v}_i, \boldsymbol{w}_{k'})}{2\sigma^2}\right)}}}_{=P(k|\boldsymbol{v}_i)} + \sum_k (\boldsymbol{c}_i - \boldsymbol{y}_k)^2 \frac{e^{\left(-\frac{d(\boldsymbol{v}_i, \boldsymbol{w}_k}{2\sigma^2}\right)}}{\sum_{k'} e^{\left(\frac{-d(\boldsymbol{v}_i, \boldsymbol{w}_{k'})}{2\sigma^2}\right)}} \\
&= \underbrace{\sum_k (\boldsymbol{c}_i - \boldsymbol{y}_j)^2 P(k|\boldsymbol{v}_i)}_{=lc_i} + (\boldsymbol{c}_i - \boldsymbol{y}_k)^2 \sum_k \frac{e^{\left(-\frac{d(\boldsymbol{v}_i, \boldsymbol{w}_k}{2\sigma^2}\right)}}{\sum_{k'} e^{\left(\frac{-d(\boldsymbol{v}_i, \boldsymbol{w}_{k'})}{2\sigma^2}\right)}} \\
&= lc_i + (\boldsymbol{c}_i - \boldsymbol{y}_k)^2 \underbrace{\sum_k \frac{e^{\left(-\frac{d(\boldsymbol{v}_i, \boldsymbol{w}_k}{2\sigma^2}\right)}}{\sum_{k'} e^{\left(\frac{-d(\boldsymbol{v}_i, \boldsymbol{w}_{k'})}{2\sigma^2}\right)}}}_{=1,\ since\ probability} \\
&= lc_i + (\boldsymbol{c}_i - \boldsymbol{y}_k)^2 \qquad (4.78)
\end{aligned}
$$

Substituting eq. (4.78) back into eq. (4.77) yields

$$
\begin{aligned}
T &= \left(lc_i + (\boldsymbol{c}_i - \boldsymbol{y}_k)^2\right) \cdot \Pi(\boldsymbol{v}_i, \boldsymbol{c}_i, \boldsymbol{w}_j, \boldsymbol{y}_j) \cdot ((\boldsymbol{c}_i - \boldsymbol{y}_j)^2 - lc_i) \\
&= \left(lc_i + (\boldsymbol{c}_i - \boldsymbol{y}_k)^2\right) \cdot ((\boldsymbol{c}_i - \boldsymbol{y}_j)^2 - lc_i) \cdot \Pi(\boldsymbol{v}_i, \boldsymbol{c}_i, \boldsymbol{w}_j, \boldsymbol{y}_j) \\
&= \underbrace{\left((\boldsymbol{c}_i - \boldsymbol{y}_k)^4 - lc_i^2\right)}_{T_0} \cdot \Pi(\boldsymbol{v}_i, \boldsymbol{c}_i, \boldsymbol{w}_j, \boldsymbol{y}_j) \\
&= T_0 \cdot \Pi(\boldsymbol{v}_i, \boldsymbol{c}_i, \boldsymbol{w}_j, \boldsymbol{y}_j) \qquad (4.79)
\end{aligned}
$$

4.B Derivation of the general FRSLVQ update rule

The learning rules for FRSLVQ (see section 4.1.2), a variant of RSLVQ incorporating fuzzy class labels, are derived in the style of the derivation approach given in (Schneider et al. 2008). There the derivative of the RSLVQ likelihood ratio (2.54) with respect to a general parameter $\Theta_j \neq \boldsymbol{v}$ was deduced. The parameter Θ_j can later on be substituted by the prototype $\boldsymbol{w}_j$, the hyper parameter σ_j^2 or another metric parameter λ_j to obtain the respective learning rule.

For further considerations the conditional density was chosen to have the normalized exponential form $p(\boldsymbol{v}|i) = K(j)e^{f(\cdots)}$ where $f(...)$ is an abbreviation for the dissimilarity function $f(\boldsymbol{v}, \boldsymbol{w}_j, \sigma_j^2, \lambda_j)$ and $K(j)$ the normalization factor. This factor depends on the shape of component i. Assuming a N-dimensional Gaussian distribution $K(j)$ can be set to $K(j) = 2\pi\sigma_j^{2(-N/2)}$.

$$
\begin{aligned}
\frac{\partial \log \frac{p(\boldsymbol{v},\boldsymbol{c}|W)}{p(\boldsymbol{v}|W)}}{\partial \Theta_j} &= \frac{\partial \log p(\boldsymbol{v},\boldsymbol{c}|W)}{\partial \Theta_j} - \frac{\partial \log p(\boldsymbol{v}|W)}{\partial \Theta_j} \\
&= \frac{1}{p(\boldsymbol{v},\boldsymbol{c}|W)} \underbrace{\frac{\partial p(\boldsymbol{v},\boldsymbol{c}|W)}{\partial \Theta_j}}_{(a)} - \frac{1}{p(\boldsymbol{v}|W)} \underbrace{\frac{\partial p(\boldsymbol{v}|W)}{\partial \Theta_j}}_{(b)} \\
&= \frac{\boldsymbol{c}^{c(\boldsymbol{w}_j)} P(j) e^{f(\cdots)}}{p(\boldsymbol{v},\boldsymbol{c}|W)} \left(\frac{\partial K(j)}{\partial \Theta_j} + K(j) \frac{\partial f(...)}{\partial \Theta_j} \right) \\
&\quad - \frac{P(j) e^{f(\cdots)}}{p(\boldsymbol{v}|W)} \left(\frac{\partial K(j)}{\partial \Theta_j} + K(j) \frac{\partial f(...)}{\partial \Theta_j} \right) \\
&= y^{c(\boldsymbol{w}_j)} \left(\frac{P(j) e^{f(\cdots)}}{p(\boldsymbol{v},\boldsymbol{c}|W)} \frac{\partial K(j)}{\partial \Theta_j} + \frac{P(j) K(j) e^{f(\cdots)}}{p(\boldsymbol{v},\boldsymbol{c}|W)} \frac{\partial f(...)}{\partial \Theta_j} \right) \\
&\quad - \left(\frac{P(j) e^{f(\cdots)}}{p(\boldsymbol{v}|W)} \frac{\partial K(j)}{\partial \Theta_j} + \frac{P(j) K(j) e^{f(\cdots)}}{p(\boldsymbol{v}|W)} \frac{\partial f(...)}{\partial \Theta_j} \right)
\end{aligned}
$$

(continued on next page)

$$
\begin{aligned}
= \quad & \frac{1}{K(j)} \underbrace{\frac{y^{c(\boldsymbol{w}_j)} P(j) K(j) e^{f(\dots)}}{p(\boldsymbol{v}, \boldsymbol{c}|W)}}_{P_{\boldsymbol{c}}(j|\boldsymbol{v})} \frac{\partial K(j)}{\partial \Theta_j} \\
& + \underbrace{\frac{y^{c(\boldsymbol{w}_j)} P(j) K(j) e^{f(\dots)}}{p(\boldsymbol{v}, \boldsymbol{c}|W)}}_{P_{\boldsymbol{c}}(j|\boldsymbol{v})} \frac{\partial f(\dots)}{\partial \Theta_j} \\
& - \frac{1}{K(j)} \underbrace{\frac{P(j) K(j) e^{f(\dots)}}{p(\boldsymbol{v}|W)}}_{P(j|\boldsymbol{v})} \frac{\partial K(j)}{\partial \Theta_j} \\
& - \underbrace{\frac{P(j) K(j) e^{f(\dots)}}{p(\boldsymbol{v}|W)}}_{P(j|\boldsymbol{v})} \frac{\partial f(\dots)}{\partial \Theta_j} \\
= \quad & P_{\boldsymbol{c}}(j|\boldsymbol{v}) \left(\frac{1}{K(j)} \frac{\partial K(j)}{\partial \Theta_j} + \frac{\partial f(\dots)}{\partial \Theta_j} \right) \\
& - P(j|\boldsymbol{v}) \left(\frac{1}{K(j)} \frac{\partial K(j)}{\partial \Theta_j} + \frac{\partial f(\dots)}{\partial \Theta_j} \right) \\
= \quad & (P_{\boldsymbol{c}}(j|\boldsymbol{v}) - P(j|\boldsymbol{v})) \left(\frac{1}{K(j)} \frac{\partial K(j)}{\partial \Theta_j} + \frac{\partial f(\dots)}{\partial \Theta_j} \right)
\end{aligned}
$$

(a)

$$\begin{aligned}
\frac{\partial p(\boldsymbol{v}, \boldsymbol{c}|W)}{\partial \Theta_j} &= \frac{\partial}{\partial \Theta_j} \left(\sum_{k=1}^{C} \boldsymbol{c}^k \sum_{j=1}^{m} \delta_{k,c(\boldsymbol{w}_j)} p(\boldsymbol{v}|j) P(j) \right) \\
&= \sum_{k=1}^{C} \boldsymbol{c}^k \sum_{j=1}^{m} \delta_{k,c(\boldsymbol{w}_j)} P(j) \frac{\partial p(\boldsymbol{v}|j)}{\partial \Theta_j} \\
&= \sum_{k=1}^{C} \boldsymbol{c}^k \sum_{j=1}^{m} \delta_{k,c(\boldsymbol{w}_j)} P(j) e^{f(\ldots)} \left(\frac{\partial K(j)}{\partial \Theta_j} + K(j) \frac{\partial f(\ldots)}{\partial \Theta_j} \right) \\
&= \boldsymbol{c}^{c(\boldsymbol{w}_j)} P(j) e^{f(\ldots)} \left(\frac{\partial K(j)}{\partial \Theta_j} + K(j) \frac{\partial f(\ldots)}{\partial \Theta_j} \right)
\end{aligned}$$

(b)

$$\begin{aligned}
\frac{\partial p(\boldsymbol{v}|W)}{\partial \Theta_j} &= \frac{\partial \sum_{j=1}^{m} p(\boldsymbol{v}|j) P(j)}{\partial \Theta_j} \\
&= P(j) \frac{\partial p(\boldsymbol{v}|i)}{\partial \Theta_j} \\
&= P(j) e^{f(\ldots)} \left(\frac{\partial K(j)}{\partial \Theta_j} + K(j) \frac{\partial f(\ldots)}{\partial \Theta_j} \right)
\end{aligned}$$

The terms $P_{\boldsymbol{c}}(j|\boldsymbol{v})$ and $P(j|\boldsymbol{v})$ are assignment probabilities. $P(j|\boldsymbol{v})$ is the assignment probability to component i independent of the class membership:

$$\begin{aligned}
P(j|\boldsymbol{v}) &= \frac{P(j) K(j) e^{f(\ldots)}}{\sum_{j=1}^{m} P(j) K(j) e^{f(\ldots)}} \\
&= \frac{P(j) K(j) e^{f(\ldots)}}{p(\boldsymbol{v}|W)}
\end{aligned}$$

$P_{\boldsymbol{c}}(j|\boldsymbol{v})$ is the assignment probability to component i within class $c(\boldsymbol{w}_j)$:

$$\begin{aligned} P_{\boldsymbol{c}}(j|\boldsymbol{v}) &= \frac{\sum_{k=1}^{C} y^k \delta_{k,c(\boldsymbol{w}_j)} P(j) K(j) e^{f(\ldots)}}{\sum_{k=1}^{C} y^k \sum_{j=1}^{m} \delta_{k,c(\boldsymbol{w}_j)} P(j) K(j) e^{f(\ldots)}} \\ &= \frac{y^{c(\boldsymbol{w}_j)} P(j) K(j) e^{f(\ldots)}}{\sum_{k=1}^{C} y^k \sum_{j=1}^{m} \delta_{k,c(\boldsymbol{w}_j)} P(j) K(j) e^{f(\ldots)}} \\ &= \frac{y^{c(\boldsymbol{w}_j)} P(j) K(j) e^{f(\ldots)}}{p(\boldsymbol{v}, \boldsymbol{c}|W)} \end{aligned}$$

Chapter 5

Summary

In this thesis known prototype based cluster and classification algorithms from the fields of machine learning and artificial neural networks are extended to incorporate fuzziness. Thereby it has to be differentiated between fuzzy clustering and fuzzy classification. The difference consists in the handling of the data. Fuzzy clustering is applied to group unlabeled data, which is hard to separate due to overlaps in the feature space. The fuzzy cluster solution allows to specify to which degree a data sample is assigned to a certain cluster. Fuzzy classification is based on fuzzy labeled training data, which means, that the learning process itself is incorporating a certain amount of uncertainty. Subsequent classification of new data samples also results in probabilistic class assignments.

In the following, a short overview of all proposed algorithms and their special features is presented. Further details can be found in the respective chapters and in the there referenced publications.

Relevance Fuzzy c-Means (R-FCM) The Fuzzy c-Means algorithm as proposed by DUNN and BEZDEK is modified to incorporate a relevance parameter. This parameter, which is adapted parallely to the prototypes and their assignments, improves the clustering in terms of separation and compactness. The improvement is verified by several measure like the Xie-Beni-Index and Fukuyama-Seguno-Index. Note, that the relevance parameter adaption has to be performed very carefully, since the metric of the distance itself is changed. Further, it is shown, that divergences can easily be applied instead of the commonly preferred Euclidean distance, which is especially useful for functional data.

Median Fuzzy c-Means (M-FCM) This algorithm combines the fuzziness from Fuzzy c-Means with the ability to handle non-vectorial data inherent in Median c-Means by BEZDEK. The prototypes of the resulting algorithm are restricted to be data samples themselves, yet allow a probabilistic clustering of the data set. M-FCM is sensitive to the prototype initialization and converges to a local minimum only, if the data set consist to a certain amount of overlapping data.

Fuzzy Affinity Propagation (FAP) Affinity propagation as proposed by FREY & DUECK is also a median clustering algorithm, yet contrary to M-FCM, not all similarities between the data points have to be given nor do they have to be symmetric. To obtain a fuzzy version, the similarities and responsibilities peculiar to Affinity Propagation are reinterpreted to allow probabilistic clustering. As common for all median clustering algorithms, the exemplars, i. e. prototypes, are data points themselves.

Fuzzy labeled FSNPC (F-FSNPC) VILLMANN ET AL. introduced an modification of SNPC by SEO ET AL., where the prototypes receive probabilistic class labels to make fuzzy class assignments possible. Yet, this as FSNPC known algorithm is restricted to crisp labeled data. The version presented in this thesis enables the algorithm to work with crisp as well as fuzzy labeled data by incorporating the fuzzy class assignments into the learning process. The analysis of the algorithm shows, that the prototypes behave differently compared to SNPC based on crisp labeled data. Instead of moving in the direction of the descision boundary, they tend to move away from it into regions of higher classification agreement.

Fuzzy Robust Soft LVQ (FRSLVQ) The Robus Soft LVQ as proposed by SEO & OBERMAYER is based on a likelihood function incorporating probability densities. Yet for the learning only crisp data sets can used. The here introduced modification replaces the crisp prototype class labels with a fuzzy labels taking the respective assignment probability to all classes into account. Further, also the training data set is allowed to contain fuzzy labeled data samples. As observed before for the F-FSNPC, the prototypes move away from the decision boundary.

Fuzzy Labeled Neural Gas (FLNG) Introducing fuzzy labels in the Neural Gas algorithm proposed by MARTINETZ allows to perform fuzzy classification although the original Neural Gas is a clustering algorithm only. To be able to consider the class labels, the cost function has to be modified by adding an additional term concerning the fuzzy labels. Three different version for the prototype updates and their class labels are presented, depending on the type of the data – discrete or continuous – and the chosen neighborhood function. Further is shown that relevance learning can easily be incorporated to identify relevant features.

Fuzzy Labeled SOM (FLSOM) Self Organizing Maps as introduced by KOHONEN employ an unsupervised learning scheme. Modifications by HESKES led to the formulation of a cost function, which further allowed to incorporate class information. The here presented Fuzzy Labeled SOM is an extension thereof incorporating fuzzy

labeled data. Similar to FLNG, an additional term taking the fuzzy class labels into account is added to the cost function. The derivation of the update rules is based on a general distance measure, which can easily be adapted to incorporate relevance learning.

A further aspect of this thesis are considerations concerning evaluation measures for classifications and cluster solution. The used measures to evaluate clusterings based on separation and compactness, like Xie-Beni-Index and Fukuyama-Sugeno-Index, are only useful for comparison, if the *correct* number of clusters is chosen. Yet usually, this number is not known in advance. For the purpose of this thesis, where the used data sets are known and formerly investigated, the number of prototypes was set appropriately. But in general, dealing with new data sets, the mentioned indexes are not satisfying.

For the evaluation of classifiers Cohen's Kappa index (for two classifiers) and Fleiss' Kappa index (for more than two classifiers) are applied. To evaluate fuzzy classifications a variant of Cohen's Kappa exists. The respective Fuzzy Fleiss' Kappa is introduced in section 2.4.3. Although the interpretation of the index values in terms of *perfect* or *moderate* agreement and so on is controversial, the measure is still useful to objectively evaluate the performance of an algorithm compared to another algorithm or a reference solution. Under certain conditions the index can also be applied to (fuzzy) cluster solutions.

Outlook

The proposed Fuzzy Labeled Neural Gas algorithm is a semi-supervised learning scheme incorporating fuzzy class labels. Recently, we developed the unsupervised Fuzzy Neural Gas by modifying the FCM cost function. By replacing the distances used to calculate the costs with a local cost function as known from NG, an unsupervised learning scheme based on neighborhood rankings is obtained. As of now the proposed algorithm is published as a technical report. A presentation including examples and performance results is in work.

As mentioned before, it is difficult to evaluate a cluster solution. Either reference solutions have to be available as requested by the Rand-Index or Kappa-Indexes, or measures based on separation and compactness like Xie-Beni-Index and Fukuyama-Sugeno-Index have to be considered. A promising validation method for crisp clusterings has been proposed by TAŞDEMIR & MERÉNYI, where the inter and intra connectivity of the clusters is taken into account. During the last months, yet not within the scope of this thesis, we modified this measure to evaluate fuzzy cluster-

ings. First results have been presented at the ESANN 2012 in Brugge this year, but further work is still necessary to completely understand and verify this measure.

Theses

1. Enabling established algorithms to incorporate fuzziness results in fuzzy cluster solutions or fuzzy classifiers delivering an intuitively more natural presentation of the data. Validating this assumption is a difficult task.

2. The Euclidean distance is not the one and only. Depending on the specific problem and considering the nature of the data other dissimilarity measures, e. g. divergences, might be more suitable.

3. Validating a (fuzzy) cluster solution is an ill-posed problem in general. Measures based on separation and compactness only work well, if the *right* number of prototypes is chosen.

4. Relevance clustering improves the separability and compactness of a clustering by enforcing the influence of certain dimensions.

5. Fuzzy classification works even if the training data set has mainly crisp class labels.

6. Incorporating relevance learning improves the fuzzy classification results.

7. The meaning of the word *fuzzy* has become increasingly fuzzy.

8. There are always two sides of a coin ... or three ... or four ...

9. *Fuzzy* is just a word.

10. Nur Mut!

Publications

Journal Papers

[1] T. Villmann, T. Geweniger, M. Kästner, and M. Lange. Fuzzy neural gas for unsupervised vector quantization. *Artificial Intelligence and Soft Computing – Lecture Notes in Computer Science*, 7267: 350-358. Springer Berlin/Heidelberg, 2012.

[2] T. Geweniger, D. Zühlke, B. Hammer, and T. Villmann. Median fuzzy c-means for clustering dissimilarity data. *Neurocomputing*, 73 (7-9): 1109-1116, 2010.

[3] T. Villmann, B. Hammer, F.-M. Schleif, T. Geweniger, and W. Herrmann. Fuzzy classification by fuzzy labeled neural gas. *Neural Networks*, 19: 772-779, 2006.

Conference Papers

[1] T. Geweniger, M. Kästner, M. Lange, and T. Villmann. Modified conn-index for the evaluation of fuzzy clusterings. *Proceedings of European Symposium on Artificial Neural Networks (ESANN'2012)*, Belgium, 2012.

[2] T. Geweniger and T. Villmann. Extending FSNPC to handle data points with fuzzy class assignments. *Proceedings of European Symposium on Artificial Neural Networks (ESANN'2010)*, Belgium, 2010.

[3] D. Zühlke, F.-M. Schleif, T. Geweniger, S. Haase, and T. Villmann. Learning vector quantiziation for heterogeneous structured data. *Proceedings of European Symposium on Artificial Neural Networks (ESANN'2010)*, Belgium, 2010.

[4] T. Geweniger, D. Zühlke, B. Hammer, and T. Villmann. Median variant of fuzzy c-means. *Proceedings of European Symposium on Artificial Neural Networks (ESANN'2009)*, Belgium, 2009.

[5] T. Geweniger, P. Schneider, and T. Villmann. Erweiterung des RSLVQ-Algorithmus zur Berechnung unscharfer Klassenzugehörigkeiten. *Scientific Reports - Journal of the University of Applied Sciences Mittweida*, 6: 18-21, 2009.

[6] T. Geweniger, D. Zühlke, B. Hammer, and T. Villmann. Fuzzy variant of affinity propagation in comparison to median fuzzy c-means. *Advances in Self-Organizing Maps – Proceeding of the Workshop on Self-Oranizing Maps (WSOM)*, LNCS 5629: 72-79. Springer, 2009.

[7] D. Zühlke, T. Geweniger, U. Heimann, and T. Villmann. Fuzzy Fleiss-Kappa for comparison of fuzzy classifiers. *Proceedings of European Symposium on Artificial Neural Networks (ESANN'2009)*, pages 269-274, Belgium, 2009.

[8] T. Geweniger, F.-M. Schleif, A. Hasenfuss, B. Hammer, and T. Villmann. Comparison of cluster algorithms for the analysis of text data using Kolmogorov complexity. *Proceedings of the International Conference on Neural Information Processing (ICONIP'2008)*, pages 61-69. Springer, 2009.

[9] A. Hasenfuss, B. Hammer, T. Geweniger, and T. Villmann. Magnification control in relational neural gas. *Proceedings of European Symposium on Artificial Neural Networks (ESANN'2008)*, Belgium, 2008.

[10] T. Villmann, U. Seiffert, F.-M. Schleif, C. Brüß, T. Geweniger, and B. Hammer. Fuzzy labeled self-organizing map with label-adjusted prototypes. *Proceedings of Conference Artificial Neural Networks in Pattern Recognition (ANNPR) 2006*, Germany, LNAI 4087: 46-56. Springer, 2006.

[11] T. Villmann, B. Hammer, F.-M. Schleif, T. Geweniger, T. Fischer, and M. Cottrell. Prototype based classification using information theoretic learning. *Proceedings of International Conference on Neural Information Processing (ICONIP), Hongkong 2006*, volume II, pages 40-49, Springer Heidelberg/New York, 2006.

[12] T. Villmann, B. Hammer, F.-M. Schleif, and T. Geweniger. Fuzzy labeled neural gas for fuzzy classification. *Proceedings of Workshop on Self-Organizing Maps (WSOM) 2005*, pages 283-290, 2005.

[13] G. Strauss, C. Trantakis, D. Winkler, T. Geweniger, M. Bublat, T. Schulz, M. Buecheler, and F. Bootz. Enhanced CAS in head and neck-surgery: evaluation of intraoperative data modality IMRI and sonography. *International Congress Series – Computer Assisted Radiology and Surgery.*, 1230: 14-19, 2001.

Technical Reports

[1] T. Geweniger, M. Kästner, M. Lange, and T. Villmann. Derivation of a generalized conn-index for fuzzy clustering validation. In T. Villmann and F.-M. Schleif, editors, *Machine Learning Reports 07/2011*, pages 1-10, 2011.

[2] T. Villmann, T. Geweniger, M. Kästner, and M. Lange. Theory of fuzzy neural gas for unsupervised vector quantization. *Machine Learning Reports 06/2011, MiWoCI 2011*, 2011.

[3] T. Geweniger, P. Schneider, F.-M. Schleif, M. Biehl, and T. Villmann. Extending RSLVQ to handle data points with fuzzy class assignments. *Machine Learning Report 02/2009*, University of Leipzig, 2009.

[4] T. Villmann, B. Hammer, F.-M. Schleif, T. Geweniger, and M. Cottrell. Fuzzy learning vector quantization by density matching. *Technical report, Clausthal University of Technology, Institute of Computer Science*, Clausthal-Zellerfeld, Germany, 2006.

Book Chapters

[1] T. Villmann, T. Geweniger, B. Bergmann, and A. Gumz. Soziophysiologie von Therapieprozessen - die Beziehung zwischen Therapeut, Patient und gesprochenem Wort. In G. Schiepek, editor, *Neurobiologie der Psychotherapie* 2nd ed., pp. 350-364, Schattauer, 2010.

Bibliography

Albani, C., Reulecke, M., Körner, A., Villmann, T., Blaser, G., Geyer, M., Pokorny, D. and Kächele, H.: 2002, Erinnertes elterliches Erziehungsverhalten und zentrale Beziehungsmuster bei Psychotherapiepatientinnen, *Psychotherapie Forum* **9**(4), 162–171.

Alex, N., Hasenfuss, A. and Hammer, B.: 2009, Patch clustering for massive data sets, *Neurocomputing* **72**(7-9), 1455–1469.

Arnonkijpanich, B., Hasenfuss, A. and Hammer, B.: 2010, Local matrix learning in clustering and applications for manifold visualization, *Neural Networks* **23**(4), 476–486.

Aronszajn, N.: 1950, Theory of reproducing kernels, *Transactions of the American Mathematical Society* **68**, 337–404.

Ball, G. H. and Hall, D. J.: 1965, Isodata, a novel method of data analysis and pattern classification, *Technical Report AD699616*, Stanford Research Institute.

Ball, G. H. and Hall, D. J.: 1967, A clustering technique for summarizing multivariate data, *Behavioral Science* **12**(2), 153–155.

Bennett, C. H., Gács, P., Li, M., Vitányi, P. M. B. and Zurek, W. H.: 1998, Information distance, *IEEE Transactions on Information Theory* **44**(4).

Bezdek, J.: 1980, A convergence theorem for the fuzzy isodata clustering algorithms, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2**(1), 1–8.

Bezdek, J. C.: 1974a, Cluster validity with fuzzy sets, *Journal of Cybernetics* **3**, 58–72.

Bezdek, J. C.: 1974b, Numerical taxonomy with fuzzy sets, *Journal of Mathematics and Biology* **1**, 57–71.

Bezdek, J. C.: 1981, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York.

Bezdek, J. C., Hathaway, R. J., Huband, J. M., Leckie, C. and Kotagiri, R.: 2006, Approximate clustering in very large relational data, *International Journal of Intelligent Systems* **21**(8), 817–841.

Bezdek, J. C., Tsao, E. C. K. and Pal, N. R.: 1992, Fuzzy Kohonen clustering networks, *Fuzzy Systems, 1992., IEEE International Conference on*, pp. 1035 –1043.

Boulet, R., Jouve, B., Rossi, F. and Villa, N.: 2008, Batch kernel SOM and related Laplacian methods for social network analysis, *Neurocomputing* **71**(7-9), 1257–1273.

Brüß, C., Bollenbeck, F., Schleif, F.-M., Weschke, W., Villmann, T. and Seiffert, U.: 2006, Fuzzy image segmentation with fuzzy labeled neural gas, *ESANN 2006 proceedings - European Symposium on Artificial Neural Networks*, d-side publications, Bruges (Belgium), pp. 563–568.

Campello, R.: 2007, A fuzzy extension of the rand index and other related indexes for clustering and classification assessment, *Pattern recognition Letters 28* .

Cannon, R., Dave, J. and Bezdek, J. C.: 1986, Efficient implementation of the fuzzy c-means clustering algorithms, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **8**(2), 248–255.

Cichocki, A. and Amari, S.: 2010, Families of alpha- beta- and gamma- divergences: Flexible and robust measures of similarities, *Entropy* **12**(6), 1532–1568.

Cichocki, A., Zdunek, R., Phan, A.-H. and Amari, S.: 2009, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis*, John Wiley.

Cilibrasi, R. and Vitányi, P. M. B.: 2005, Clustering by compression, *IEEE Transactions on Information Theory* **51**(4), 1523–1545.

Cohen, J.: 1960, A coefficient of agreement for nominal scales, *Educational and Psychological Measurement* **20**, 37–46.

Cottrell, M., Hammer, B., Hasenfuss, A. and Villmann, T.: 2006, Batch and median neural gas, *Neural Networks* **19**, 762–771.

Cottrell, M., Ibbou, S. and Letrémy, P.: 2004, SOM-based algorithms for qualitative variables, *Neural Networks* **17**, 1149–1168.

Crammer, K., Gilad-Bachrach, R., Navot, A. and Tishby, A.: 2002, Margin analysis of the LVQ algorithm, *Proc. NIPS 2002*, http://www-2.cs.cmu.edu/Groups/NIPS/NIPS2002/NIPS2002preproceedings/index.html.

Dave, R. N.: 1990, Fuzzy shell-clustering and application to circle detection in digital images, *International Journal of General Systems* **16**, 343–355.

de Oliveira, J. V. and Pedrycz, W. (eds): 2007, *Advances in Fuzzy Clustering and its Applications*, John Wiley and Sons Inc.

de Wouwer, G. V., Scheunders, P., Dyck, D. V., Bodt, M. D., Wuyst, F. and de Heyning, P. V.: 1996, Wavelet-FILVQ classifier for speech analysis, *Proceedings of the International Conference on Pattern Recognition*, pp. 214–218.

Denoeux, T. and Masson, M.-H.: 2004, EVCLUS: evidential clustering of proximity data, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* **34**(1), 95–109.

Dou, W., Ren, Y., Wu, Q., Ruan, S., Chen, Y., Bloyet, D. and Constans, J.-M.: 2007, Fuzzy kappa for the agreement measure of fuzzy classifications, *Neurocomputing* **70**, 726–734.

Duda, R. and Hart, P.: 1973, *Pattern Classification and Scene Analysis*, Wiley, New York.

Dunn, J. C.: 1973, A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters, *Journal of Cybernetics* **3**(3), 32–57.

Eguchi, S. and Kano, Y.: 2001, Robustifying maximum likelihood estimation, *Technical Report 802*, Tokyo Institute of Statistical Mathematics.

Emmert-Streib, F. and Dehmer, M. (eds): 2009, *Information Theory and Statistical Learning*, Springer.

Fisher, P. S., Baek, J., Adeyeye, J. O. and Setubal, J.: 2010, Finite inductive sequences, Kolmogorov complexity with application to genome sequences, *in* M. Doble, W. Loging and Z. Sun (eds), *International Conference on Bioinformatics, Computational Biology, Genomics and Chemoinformatics (BCBGC-10)*, ISRST, Orlando, Florida, USA, pp. 78–83.

Fleiss, J. L.: 1981, *Statistical Methods for Rates and Proportions*, 2nd edn, Wiley, New York.

Fowlkes, E. B. and Mallows, C. L.: 1983, A method for comparing two hierarchical clusterings, *Journal of the American Statistical Association* **78**(383), 553–569.

Fréchet, M.: 1906, Sur quelques points du calcul fonctionnel, *Rendiconti del Circolo Matematico di Palermo* **22**(1), 1–72.

Frey, B. J. and Dueck, D.: 2007, Clustering by passing messages between data points, *Science* **315**, 972–976.

Fritzke, B.: 1995, A growing neural gas network learns topologies, *Advances in Neural Information Processing Systems 7*, MIT Press, pp. 625–632.

Fujisawa, H. and Eguchi, S.: 2008, Robust parameter estimation with a small bias against heavy contamination, *Journal of Multivariate Analysis* pp. 2053–2081.

Fukuyama, Y. and Sugeno, M.: 1989, A new method of choosing the number of clusters for the fuzzy c-means method, *Proc. of the 5th Fuzzy Systems Symposium* pp. 247–250.

Gath, I. and Geva, A.: 1989, Unsupervised optimal fuzzy clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **11**, 773–791.

Geweniger, T., Kästner, M. and Villmann, T.: 2011, Optimization of parametrized divergences in fuzzy c-means, *in* M. Verleysen (ed.), *Proc. of European Symposium on Artificial Neural Networks (ESANN'2011)*, d-side publications, Evere, Belgium, pp. 11–16.

Geweniger, T., Schleif, F.-M., Hasenfuss, A., Hammer, B. and Villmann, T.: 2009, Comparison of cluster algorithms for the analysis of text data using kolmogorov complexity, *Advances in Neuro-Information Processing: 15th International Conference (ICONIP 2008), Auckland, New Zealand, November 25-28, 2008, Revised Selected Papers, Part II*, Springer-Verlag, Berlin, Heidelberg, pp. 61–69.

Geweniger, T., Schneider, P., Schleif, F.-M., Biehl, M. and Villmann, T.: 2009, Extending RSLVQ to handle data points with uncertain class assignments, *Machine Learning Report 2*, University of Leipzig Working Group Computational Intelligence.

Geweniger, T. and Villmann, T.: 2010, Extending FSNPC to handle data points with fuzzy class assignments, *in* M. Verleysen (ed.), *Proc. of European Symposium on Artificial Neural Networks (ESANN'2010)*, d-side publications, Brussels, Belgium.

Geweniger, T., Zühlke, D., Hammer, B. and Villmann, T.: 2009, Fuzzy variant of affinity propagation in comparison to median fuzzy c-means, *in* J. Principe (ed.), *Advances in Self-Organizing Maps - Proceeding of the Workshop on Self-Oranizing Maps (WSOM)*, LNCS 5629, Springer, pp. 72–79.

Geweniger, T., Zühlke, D., Hammer, B. and Villmann, T.: 2010, Median fuzzy c-means for clustering dissimilarity data, *Neurocomputing* **73**(7-9), 1109–1116.

Gordon, A. D.: 1998, How many clusters? An investigation of five procedures for detecting nested cluster structures, *in* C. H. et Al. (ed.), *Data Science, Classification and Related Methods*, Springer, Tokyo - New York - Berlin, pp. 109–116.

Graepel, T., Burger, M. and Obermayer, K.: 1997, Deterministic annealing for topographic vector quantization and self-organizing maps, *Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland, June 4–6*, Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, pp. 345–350.

Gustafson, E. and Kessel, W.: 1979, Fuzzy clustering with a fuzzy covariance matrix, *IEEE CDC*, IEEE Society, San Diego, California, pp. 761–766.

Hammer, B. and Hasenfuss, A.: 2007, Relational neural gas, *in* J. Hertzberg, M. Beetz and R. Englert (eds), *KI 2007: Advances in Artificial Intelligence*, LNAI 4667, Springer, Berlin Heidelberg, pp. 190–204.

Hammer, B., Hasenfuss, A., Schleif, F.-M. and Villmann, T.: 2009, Supervised median clustering, *IFI Technical Report Series IFI-06-09*, Department of Informatics Clausthal University of Technology.

Hammer, B., Strickert, M. and Villmann, T.: 2005, Supervised neural gas with general similarity measure, *Neural Processing Letters* **21**, 21–44.

Hammer, B. and Villmann, T.: 2002, Generalized relevance learning vector quantization, *Neural Networks* **15**, 1059–1068.

Hammer, B. and Villmann, T.: 2007, How to process uncertainty in machine learning?., *in* M. Verleysen (ed.), *Proc. of European Symposium on Artificial Neural Networks (ESANN'2007)*, d-side publications, Evere, Belgium, pp. 79–90.

Hasenfuss, A., Hammer, B., Geweniger, T. and Villmann, T.: 2008, Magnification control in relational neural gas, *ESANN*.

Hastie, T. and Stuetzle, W.: 1989, Principal curves hastie, *Journal of the American Statistical Association* **84**(406), 502–516.

Hathaway, R. and Bezdek, J.: 1986, Local convergence of the fuzzy c-means algorithm, *Pattern recognition* **19**(6), 477–480.

Hathaway, R. and Bezdek, J.: 1994, NERF c-means: Non-Euclidean relational fuzzy clustering, *Pattern recognition* **27**(3), 429–437.

Hathaway, R., Davenport, J. and Bezdek, J.: 1989, Relational duals of the c-means clustering algorithms, *Pattern recognition* **22**(2), 205–212.

Haykin, S.: 1999, *Neural Networks - A Comprehensive Foundation*, 2nd edn, Prentice Hall.

Hecht-Nielsen, R.: 1987, Counterpropagation networks, *Applied Optics* **26**(23), 4979–4983.

Hecht-Nielsen, R.: 1988, Applications of counterpropagation networks, *Neural Networks* **1**, 131–139.

Heskes, T.: 1999, Energy functions for self-organizing maps, *in* E. Oja and S. Kaski (eds), *Kohonen Maps*, Elsevier, Amsterdam, pp. 303–316.

Hofmann, T. and Buhmann, J. M.: 1997, Pairwise data clustering by deterministic annealing, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**(1).

Hubert, L. and Arabie, P.: 1985, Comparing partitions, *Journal of Classification* **2**, 193–218. 10.1007/BF01908075.

Hulle, M. M. V.: 2009, *Encyclopedia of Computer Science and Engineering*, Vol. 3, Wiley, chapter Kernel-based topographic maps: Theory and applications, pp. 1633–1650.

Inokuchi, R. and Miyamoto, S.: 2008, Fuzzy c-means algorithms using Kullback-Leibler Divergence and Helliger distance based on multinomial manifold, *Journal of Advanced Computational Intelligence and Intelligent Informatics* **12**(5), 443–447.

Ismail, M. and Selims, S.: 1986, Fuzzy c-means: Optimality of solutions and effictive termination of the algorithm, *Pattern recognition* **19**(6), 481–485.

Jain, A. K. and Dubes, R. C.: 1988, *Algorithms for Clustering Data*, Prentice Hall.

Kato, T.: 1950, On the adiabatic theorem of quantum mechanics, *Journal of the Physical Society of Japan* **5**(6), 435–439.

Kim, D.-W., Lee, K. H. and Lee, D.: 2004, On cluster validity index for estimation of the optimal number of fuzzy clusters, *Pattern Recognition* **37**, 2009–2025.

Kohonen, T.: 1986, Learning vector quantization for pattern recognition, *Report TKK-F-A601*, Helsinki University of Technology, Espoo, Finland.

Kohonen, T.: 1990, The self-organizing map, *Proceedings of the IEEE* **78**(9), 1464–1480.

Kohonen, T.: 1995, *Self-Organizing Maps*, Vol. 30 of *Springer Series in Information Sciences*, Springer, Berlin, Heidelberg. (Second Extended Edition 1997).

Kohonen, T. and Somervuo, P.: 2002, How to make large self-organizing maps for nonvectorial data, *Neural Networks* **15**(8-9), 945–952.

Kruse, R., Döring, C. and Lesot, M.-J.: 2007, *Advances in Fuzzy Clustering and its Applications*, John Wiley & Sons, Ltd., chapter Fundamentals of Fuzzy Clustering.

Kullback, S. and Leibler, R.: 1951, On information and sufficiency, *Annals of Mathematical Statistics* **22**, 79–86.

Kushner, H. and Clark, D.: 1978, *Stochastic Approximation Methods for Constrained and Unconstrained Systems*, Springer-Verlag, New York.

Landgrebe, D.: 2003, *Signal Theory Methods in Multispectral Remote Sensing*, Wiley, Hoboken, New Jersey.

Lehn-Schiøler, T., Hegde, A., Erdogmus, D. and Principe, J.: 2005, Vector quantization using information theoretic concepts, *Natural Computing* **4**(1), 39–51.

Li, M. and Sleep, M. R.: 2004, Melody classification using a similarity metric based on Kolmogorov complexity, *Proceedings of the Sound and Music Computing Conference (SMC'04)*, Paris, France.

Linde, Y., Buzo, A., Gray, R. and Kieffer, J.: 1979, Optimal block quantization for sources without a statistical model, *IEEE International Symposium on Information Theory*, Grignano, Italy.

Linde, Y., Buzo, A. and Gray, R. M.: 1980, An algorithm for vector quantizer design, *IEEE Transactions on Communications* **COM-28**(1), 84–95.

Luxburg, U. V.: 2007, A tutorial on spectral clustering, *Statistics and Computing* **17**(4), 395–416.

MacQueen, J. B.: 1967, Some methods for classification and analysis of multivariate observations, *in* L. M. L. Cam and J. Neyman (eds), *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1, University of California Press, pp. 281–297.

Mahalanobis, P. C.: 1930, On tests and measures of group divergence, *Journal of the Asiatic Society of Bengal* **26**, 541–588.

Martinetz, T. M., Berkovich, S. G. and Schulten, K. J.: 1993, "neural-gas" network for vector quantization and its application to time-series prediction, *IEEE Transactions on Neural Networks* **4**(4), 558–569.

Masson, M.-H. and Denoeux, T.: 2008, ECM: An evidential version of the fuzzy c-means algorithm, *Pattern recognition* **41**(4), 1384–1397.

McKay, M., Beckman, R. and Conover, W.: 1979, A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, *Technometrics* **21**(2), 239–245.

Mwebaze, E., Schneider, P., Schleif, F.-M., Aduwo, J. R., Quinn, J. A., Haase, S., Villmann, T. and Biehl, M.: 2011, Divergence-based classification in learning vector quantization, *Neurocomputing* **74**, 1429–1435.

Nielson, F. and Nock, R.: 2009, Sided and symmetrized Bregman centroids, *IEEE Transactions on Information Theory* **55**(6), 2882–2903.

Oja, E. and Lampinen, J.: 1994, Unsupervised learning for feature extraction, *in* J. M. Zurada, R. J. M. II and C. J. Robinson (eds), *Computational Intelligence Imitating Life*, IEEE Press, pp. 13–22.

Pal, N. and Bezdek, J.: 1995, On the cluster validity for the fuzzy c-means modell, *IEEE Transactions on Information Theory* **3**(3), 370–379.

Pearl, J.: 1988, *Probabilistic Reasoning in Intelligent System*, Morgan Kaufmann.

Pękalska, E. and Duin, R. P. W.: 2005, *The Dissimilarity Representation for Pattern Recognition: Foundations and Applications*, Machine Perception and Artificial Intelligence, World Scientific Publishing Company.

Principe, J. C., Ill, J. and Xu, D.: 2000, *Information theoretic learning*, Wiley, Hoboken, New Jersey, chapter Unsupervised adaptive filtering.

Qin, A. K. and Suganthan, P. N.: 2004a, Kernel neural gas algorithms with application to cluster analysis, *Proceedings of the 17th International Conference on Pattern Recognition (ICPR 2004)* **4**, 617–620.

Qin, A. K. and Suganthan, P. N.: 2004b, A novel kernel prototype-based learning algorithm, *Proceedings of the 17th International Conference on Pattern Recognition (ICPR 2004)* **4**, 621–624.

Ramsay, J. and Silverman, B.: 2006, *Functional Data Analysis*, 2nd edn, Springer Science+Media, New York.

Rand, W. M.: 1971, Objective criteria for the evaluation of clustering methods, *Journal of the American Statistical Association* **66**(336), 846–850.

Rényi, A.: 1961, On measures of entropy and information, *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press.

Rényi, A.: 1970, Probability theory, *Technical report*, Amsterdam: Noth-Holland.

Robbins, H. and Monro, S.: 1951, A stochastic approximation method, *The Annals of Mathematical Statistics* **22**(3), 400–407.

Rose, K., Gurewitz, E. and Fox, G.: 1992, Vector quantization by deterministic annealing, *IEEE Transactions on Information Theory* **38**(4), 1249–1257.

Sachs, L.: 1992, *Angewandte Statistik*, 7-th edn, Springer Verlag.

Sato, A. and Yamada, K.: 1996, Generalized learning vector quantization, *in* D. S. Touretzky, M. C. Mozer and M. E. Hasselmo (eds), *Advances in Neural Information Processing Systems 8. Proceedings of the 1995 Conference*, MIT Press, Cambridge, MA, USA, pp. 423–9.

Sato, A. and Yamada, K.: 1998, An analysis of convergence in generalized LVQ, *in* L. Niklasson, M. Bodén and T. Ziemke (eds), *Proceedings of ICANN98, the 8th International Conference on Artificial Neural Networks*, Vol. 1, Springer, London, pp. 170–176.

Schiepek, G. (ed.): 2010, *Neurobiologie der Psychotherapie*, 2nd edn, Schattauer.

Schleif, F.-M., Villmann, T., Hammer, B. and Schneider, P.: 2011, Efficient kernelized prototype based classification, *International Journal of Neural Systems* **21**(6).

Schneider, P.: 2010, *Advanced methods for prototype-based classification*, PhD thesis, Rijksuniveriteit Groningen.

Schneider, P., Biehl, M. and Hammer, B.: 2008, Matrix adaptation in discriminative vector quantization, *IFI Technical Report Series IFI-08-08*, Department of Informatics Clausthal University of Technology.

Schneider, P., Geweniger, T., Schleif, F. M., Biehl, M. and Villmann, T.: 2011, Multivariate class labeling in Robust Soft LVQ, *in* M. Verleysen (ed.), *19th European Symposium on Artificial Neural Networks (ESANN 2011)*, d-side publishing, pp. 17–22.

Schoelkopf, B. and Smola, A.: 2002, *Learning with Kernels*, MIT Press.

Schürmann, J.: 1996, *Pattern Classification*, J. Wiley and Sons Inc., New York.

Sen, S. and Dave, R.: 2000, Agglomerative model for fuzzy relational clustering (FRC), *Proceedings of the 19th International Conference of the North American Fuzzy Information Processing Society (NAFIPS), Atlanta*, IEEE Society, pp. 267–271.

Seo, S., Bode, M. and Obermayer, K.: 2003, Soft nearest prototype classification, *IEEE Transactions on Neural Networks* **14**(2).

Seo, S. and Obermayer, K.: 2003, Soft learning vector quantization, *Neural Computation* **15**, 1589–1604.

Seo, S. and Obermayer, K.: 2004, Self-organizing maps and clustering methods for matrix data, *Neural Networks* **17**, 1211–1230.

Simpson, P. K.: 1993, Fuzzy min-max neural networks - part 2: Clustering, *IEEE Transactions on Fuzzy Systems* **1**(1), 32–45.

Sokal, R. R.: 1977, Clustering and Classification: Background and Current Directions, *in* J. Van Ryzin (ed.), *Classification and Clustering*, Academic Press, New York.

Steinwart, I.: 2001, On the influence of the kernel on the consistency of support vector machines, *Journal of Machine Learning Research* **2**, 67–93.

Sutton, R. S.: 1984, *Temporal Credit Assignment in Reinforcement Learning*, PhD thesis, University of Massachusetts.

Taşdemir, K. and Merényi, E.: 2011, A validity index for prototype-based clustering of data sets with complex structures, *IEEE Trans. Systems, Man and Cybernetics, Part B* **41**(4), 1039–1053.

Theodoridis, S. and Koutroumbas, K.: 1998, *Pattern Recognition*, Academic Press.

Villmann, T., Der, R., Herrmann, M. and Martinetz, T. M.: 1997, Topology preservation in self-organizing feature maps: exact definition and measurement, *IEEE Transactions on Neural Networks* **8**(2), 256–266.

Villmann, T., Geweniger, T., Bergmann, B. and Gumz, A.: 2010, *Neurobiologie der Psychotherapie*, 2nd edn, Schattauer, chapter Soziophysiologie von Therapieprozessen - die Beziehung zwischen Therapeut, Patient und gesprochenem Wort, pp. 350–364.

Villmann, T. and Haase, S.: 2011, Divergence based vector quantization, *Neural Computation* pp. 1343–1392.

Villmann, T. and Haase, S.: 2012, A note on gradient based learning in vector quantization using differentiable kernels for hilbert and banach spaces, *Machine Learning Report 01/2012*, University of Bielefeld.

Villmann, T. and Hammer, B.: 2002, Supervised neural gas for learning vector quantization, *in* D. Polani, J. Kim and T. Martinetz (eds), *Proc. of the 5th German Workshop on Artificial Life (GWAL-5)*, Akademische Verlagsgesellschaft - infix - IOS Press, Berlin, pp. 9–16.

Villmann, T., Hammer, B., Schleif, F.-M. and Geweniger, T.: 2005, Fuzzy labeled neural gas for fuzzy classification, *in* M. Cottrell (ed.), *Proc. of Workshop on Self-Organizing Maps (WSOM) 2005*, pp. 283–290.

Villmann, T., Hammer, B., Schleif, F.-M., Geweniger, T. and Cottrell, M.: 2006, Fuzzy learning vector quantization by density matching, *Technical report*, Clausthal University of Technology, Institute of Computer Science, Clausthal-Zellerfeld, Germany.

Villmann, T., Hammer, B., Schleif, F.-M., Geweniger, T., Fischer, T. and Cottrell, M.: 2006, Prototype based classification using information theoretic learning, *in* I. King (ed.), *Proceedings of International Conference on Neural Information Processing (ICONIP), Hongkong 2006*, Vol. II, Springer-Verlag, Heidelberg, New York, pp. 40–49.

Villmann, T., Hammer, B., Schleif, F.-M., Geweniger, T. and Herrmann, W.: 2006, Fuzzy classification by fuzzy labeled neural gas, *Neural Networks* **19**(6-7), 772–779.

Villmann, T., Liebers, C., Bergmann, B., Gumz, A. and Geyer, M.: 2008, Investigation of psycho-physiological interactions between patient and therapist during a psycho-dynamic therapy and their relation to speech in terms of entropy analysis using a neural network approach, *New Ideas in Psychology* **26**, 309–325.

Villmann, T., Liebers, C. and Geyer, M.: 2003, Untersuchung der psycho-physiologischen Interaktion von Patient und Therpeut im Rahmen für psycho-dynamische Einzeltherapien und informationstheoretische Auswertung, *Psychotherapeutische Reflexionen gesellschaftlichen Wandels* pp. 305–319.

Villmann, T., Schleif, F.-M. and Hammer, B.: 2006, Prototype-based fuzzy classification with local relevance for proteomics, *Neurocomputing* **69**(16–18), 2425–2428.

Villmann, T., Seiffert, U., Schleif, F.-M., Brüß, C., Geweniger, T. and Hammer, B.: 2006, Fuzzy labeled self-organizing map with label-adjusted prototypes, *in* F. Schwenker and S. Marinai (eds), *Proceedings of Conference Artificial Neural Networks in Pattern Recognition (ANNPR) 2006, Ulm, Germany*, LNAI 4087, Springer Verlag, pp. 46–56.

Villmann, T., Strickert, M., Brüß, C., Schleif, F.-M. and Seiffert, U.: 2007, Visualization of fuzzy information in fuzzy classification for image segmentation using MDS, *in* M. Verleysen (ed.), *Proc. of European Symposium on Artificial Neural Networks (ESANN 2007)*, d-side publications, Brussels, Belgium, pp. 103–108.

Vitányi, P. M. B. and Li, M.: 2000, Minimum description length induction, Bayesianism, and Kolmogorov complexity, *IEEE Transactions on Information Theory* **46**(2).

Žalik, K. R. and Žalik, B.: 2010, Validity index for clusters of different sizes and densities, *Pattern Recognition Letters* **32**(2011), 221–234.

Windham, M.: 1985, Numerical classification of proximity data with assignment measures, *Journal of Classification* **2**, 157–192.

Xie, X. L. and Beni, G.: 1991, A validity measure for fuzzy clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13**(8), 841–847.

Yang, X., Zhang, G., Lu, J. and Ma, J.: 2011, A kernel fuzzy c-means clustering-based fuzzy support vector machine algorithm for classification problems with outliers or noises, *Fuzzy Systems, IEEE Transactions on* **19**(1), 105 –115.

Zühlke, D., Geweniger, T., Heimann, U. and Villmann, T.: 2009, Fuzzy Fleiss-Kappa for comparison of fuzzy classifiers, *in* M. Verleysen (ed.), *Proc. of European Symposium on Artificial Neural Networks (ESANN'2009)*, d-side publications, Evere, Belgium, pp. 269–274.

Zühlke, D., Schleif, F.-M., Geweniger, T., Haase, S. and Villmann, T.: 2010, Learning vector quantiziation for heterogeneous structured data, *in* M. Verleysen (ed.), *Proc. of European Symposium on Artificial Neural Networks (ESANN'2010)*, d-side publications, Evere, Belgium.

Nederlandse samenvatting

In dit proefschrift worden bestaande op prototypen gebaseerde groeperings- en classificatiealgoritmen binnen de gebieden van machinaal leren en kunstmatige neurale netwerken uitgebreid met fuzziness. Daarbij moet onderscheid gemaakt worden tussen fuzzy groeperen en fuzzy classificeren. Het verschil zit hem er in hoe er met de data omgegaan wordt. Fuzzy groepering wordt toegepast om een verzameling van ongelabelde data te groeperen, die lastig te scheiden is wegens overlappingen in de eigenschappen ruimte. De fuzzy groeperingsoplossing staat ons toe de mate waarin een datapunt wordt toegekend aan een bepaalde groep te specificeren. Bij fuzzy classificatie wordt gewerkt met fuzzy gelabelde trainingsdata, oftewel het leerproces zelf houdt al rekening met een zekere mate van onzekerheid. De daarop volgende classificatie van nieuwe datapunten resulteert eveneens in probabilistische klassentoekenningen.

Hierna wordt een kort overzicht van alle voorgestelde algoritmes en hun speciale eigenschappen gegeven. Verdere details staan in de respectievelijke hoofdstukken en de daarin gerefereerde publicaties.

Relevance Fuzzy c-Means (R-FCM) Het Fuzzy c-Means algoritme zoals voorgesteld door DUNN en BEZDEK is uitgebreid met een relevantie parameter. Deze parameter, die parallel met de prototypen en hun groepentoekenningen wordt aangepast, verbetert de groepering in termen van separatie en compactheid. Deze verbetering is geverifieerd door aan aantal maten zoals de Xie-Beni-Index en de Fukuyama-Seguno-Index. Merk op dat het aanpassen van de relevantie parameter zeer voorzichtig moet gebeuren, aangezien de metriek van de afstand zelf wordt veranderd. Daarnaast wordt aangetoond dat divergenties gemakkelijk als alternatief gebruikt kunnen worden voor de Euclidische afstandsmaat, die normaal de voorkeur geniet. Dit is in het bijzonder nuttig voor functionele data.

Median Fuzzy c-Means (M-FCM) Dit algoritme combineert de fuzziness uit Fuzzy c-Means met de inherente mogelijkheid van Median c-Means, zoals voorgesteld door BEZDEK, om om te gaan met niet-vectoriele data. De prototypes van het resulterende algoritme zijn beperkt tot de datapunten zelf, maar ze maken wel een probabilistische groepering van de data verzameling mogelijk. M-FCM is gevoelig voor de initialisatie van de prototypes en convergeert alleen naar een lokaal minimum als de data set maar een beperkte hoeveelheid overlappende data bevat.

Fuzzy Affinitiy Propagation (FAP) Affinity propagation zoals voorgesteld door FREY & DUECK is ook een median groeperingsalgoritme, maar, in tegenstelling tot M-FCM, hoeven niet alle nabijheden tussen de datapunten gegeven noch symmetrisch te zijn. Om een fuzzy variant te verkrijgen worden alle nabijheden en verantwoordelijkheden van Affinity Propagation geherinterpreteerd. Zoals gebruikelijk voor median groeperingsalgoritmes, zijn de voorbeelden, dat wil zeggen de prototypes, zelf datapunten.

Fuzzy labeled FSNPC (F-FSNPC) VILLMANN ET AL. introduceerden een variant op SNPC van SEO ET AL., waar de prototypes probabilistische klassenlabels krijgen om fuzzy klassentoekenningen mogelijk te maken. Echter, dit onder FSNPC bekende algoritme werkt alleen voor crisp gelabelde data. De in dit proefschrift gepresenteerde variant van het algoritme kan met zowel crisp als fuzzy gelabelde data omgaan doordat de fuzzy klassentoekenning is opgenomen in het leerproces. De analyse van het algoritme laat zien dat de prototypes zich anders gedragen ten opzichte van SNPC gebaseerd op crisp gelabelde data. In plaats van dat ze zich richting de beslissingsgrenzen bewegen, hebben ze de neiging daarvan weg te bewegen richting regioŠs met een hogere overeenkomst in klassentoekenningen.

Fuzzy Robust Soft LVQ (FRSLVQ) De Robust Soft LVQ gepresenteerd door SEO & OBERMAYER is gebaseerd op een aannemelijkheidsfunctie waarin de kansdichtheden zijn opgenomen. Echter, tijdens het leren kunnen alleen crispe dataverzamelingen gebruikt worden. De hier geïntroduceerde variant vervangt de crispe prototypeklassenlabels door fuzzy labels, waardoor de respectievelijke toekenningskansen tot alle klassen meegenomen wordt. Verder mag ook de trainingsdataverzameling fuzzy gelabelde datapunten bevatten. Zoals al bij F-FSNPC opgemerkt werd, bewegen de prototypes ook hier weg van de beslissingsgrenzen.

Fuzzy Labeled Neural Gas (FLNG) Door fuzzy labels toe te voegen aan het Neural Gas algoritme gepresenteerd door MARTINETZ kunnen fuzzy classificaties uitgevoerd worden, hoewel het originele Neural Gas alleen een groeperingsalgoritme

is. Om de klassenlabels in ogenschouw te kunnen nemen moet er een extra term voor de fuzzy labels aan de kostenfunctie toegevoegd worden. Drie verschillende varianten voor het updaten van de prototypes en hun klassenlabels worden gepresenteerd. Deze hangen af van het type data – discreet of continu – en de gekozen buurfunctie. Tot slot wordt aangetoond dat relevance learning op eenvoudige wijze toegevoegd kan worden om belangrijke eigenschappen te identificeren.

Fuzzy Labeled SOM (FLSOM) Self Organizing Maps zoals geïntroduceerd door KOHONEN gebruiken een ongecontroleerd leerschema. Aanpassingen door HESKES leidden tot het opstellen van een kostenfunctie, hetgeen het mogelijk maakte klasseninformatie daarin op te nemen. Het hier gepresenteerde Fuzzy Labeled SOM is een uitbreiding daarvan voor fuzzy gelabelde data. Net als bij FLNG wordt er een extra term voor de fuzzy klassenlabels toegevoegd aan de kostenfunctie. De afleiding van de updateregels is gebaseerd op een algemene afstandsmaat. Deze kan eenvoudig aangepast worden om relevance learning toe te voegen.

Een ander aspect van dit proefschrift zijn overwegingen op het gebied van evaluatiematen voor classificatie- en groeperingsoplossingen. De op separatie en compactheid gebaseerde maten om groeperingen te evalueren, zoals de Xie-Beni-Index en de Fukuyama-Sugeno-Index, kunnen alleen gebruikt worden in vergelijkingen als het *juiste* aantal groepen wordt gekozen. Normaal gesproken is dit aantal echter niet van te voren bekend. Voor dit proefschrift, waar de gebruikte dataverzamelingen bekend en al eerder onderzocht zijn, is het aantal prototypes op de gepaste waarde ingesteld. Echter, in het algemeen, wanneer er met nieuwe dataverzamelingen gewerkt wordt, voldoen de genoemde indices niet.

Voor het evalueren van classificatoren worden Cohen's Kappa index (voor twee classificatoren) en Fleiss' Kappa index (voor meer dan twee classificatoren) toegepast. Er bestaat een variant van Cohen's Kappa voor het evalueren van fuzzy classificatoren. De corresponderende Fuzzy Fleiss' kappa wordt in sectie 2.4.3 geïntroduceerd. Hoewel de interpretatie van de index in termen van *perfecte* en *gematigde* overeenstemming etc. controversieel is, is de methode nog steeds nuttig om objectief de prestaties van een algoritme te vergelijken met een andere oplossing of een referentieoplossing. Onder bepaalde voorwaarden kan de index ook toegepast worden op (fuzzy) groeperingsoplossingen.

Vooruitzicht

Het voorgestelde Fuzzy Labeled Neural Gas algoritme is een halfgecontroleerd leerschema waarin fuzzy klassenlabels zijn opgenomen. Onlangs hebben we een ongecontroleerde variant van Fuzzy Neural Gas ontwikkeld door de FCM kostenfunctie aan te passen. Een ongecontroleerde leerschema gebaseerd op buurrankingen wordt verkregen door de afstanden die gebruikt worden om de kosten te berekenen te vervangen door een lokale kostenfunctie zoals bekend van NG. Op dit moment is het voorgestelde algoritme als technisch rapport gepubliceerd. Er wordt aan een presentatie inclusief voorbeelden en prestatiemetingen gewerkt.

Zoals eerder al opgemerkt is het moeilijk om een groeperingsoplossing te evalueren. Ofwel er moeten referentie implementaties beschikbaar zijn, zoals in het geval van de Rhand-Index of de Kappa-Indices, ofwel maten gebaseerd op de separatie en compactheid zoals de Xie-Beni-Index en de Fukuyama-Sugeno-Index moeten in ogenschouw genomen worden. Een veelbelovende validatie methode voor crispe groeperingen is voorgesteld door TAŞDEMIR & MERÉNYI, waar de inter- en intragroepensamenhang in ogenschouw genomen worden. Gedurende de afgelopen maanden, maar niet binnen het kader van dit proefschrift, hebben we deze maat aangepast om ook fuzzy groeperingen te evalueren. Eerste resultaten zijn gepresenteerd op de ESANN 2012 in Brugge dit jaar, maar verder werk is zeker nodig om deze maat compleet te begrijpen en te verifiëren.

Tina Geweniger: *Fuzzy Variants of Prototype Based Clustering and Classification Algorithms*, PhD thesis, Rijksuniveriteit Groningen, 2012.

ISBN: 978-90-367-5749-2 (paper)
ISBN: 978-90-367-5748-5 (eBook)